PremiumSoft™
Premiumsoft CyberTech Ltd.

Navicat™ 11

User Guide

Table of Contents

Getting Started 8
System Requirements 9
Registration 9
Installation 10
Maintenance/Upgrade 11
End-User License Agreement 11
Connection 17
General Settings 18
Advanced Settings 21
SSL Settings 24
SSH Settings 25
HTTP Settings 26
Server Objects 28
MySQL/MariaDB Objects 28
MySQL/MariaDB Tables 28
MySQL/MariaDB Table Fields 29
MySQL/MariaDB Table Indexes 31
MySQL/MariaDB Table Foreign Keys 32
MySQL/MariaDB Table Triggers 32
MySQL/MariaDB Table Options 33
MySQL/MariaDB Views 36
MySQL/MariaDB Functions/Procedures 37
MySQL/MariaDB Events 38
Oracle Objects 40
Oracle Data Pump (Available only in Full Version) 40
Oracle Data Pump Export 40
Oracle Data Pump Import 44
Oracle Debugger (Available only in Full Version) 47
Oracle Physical Attributes/Default Storage Characteristics 49
Oracle Tables 50
Oracle Normal Tables 50
Oracle Table Fields 51
Oracle Table Indexes 52
Oracle Table Foreign Keys 53
Oracle Table Uniques 53
Oracle Table Checks 54
Oracle Table Triggers 54
Oracle Table Options 55
Oracle External Tables 56
Fields for Oracle External Tables 56
External Properties for Oracle External Tables 57
Access Parameters for Oracle External Tables 58

Oracle Index Organized Tables 58
Options for Oracle Index Organized Tables 58
Oracle Views 59
Oracle Functions/Procedures 60
Oracle Database Links 61
Oracle Indexes 62
Oracle Java 64
Oracle Materialized Views 65
Oracle Materialized View Logs 68
Oracle Packages 69
Oracle Sequences 70
Oracle Synonyms 71
Oracle Triggers 71
Oracle Types 74
Oracle XML Schemas 75
Oracle Recycle Bin 76
Oracle Directories 77
Oracle Tablespaces 77
Oracle Public Database Links 80
Oracle Public Synonyms 80
PostgreSQL Objects 80
PostgreSQL Schemas 82
PostgreSQL Tables 82
PostgreSQL Table Fields 82
PostgreSQL Table Indexes 83
PostgreSQL Table Foreign Keys 85
PostgreSQL Table Uniques 85
PostgreSQL Table Checks 86
PostgreSQL Table Excludes 86
PostgreSQL Table Rules 87
PostgreSQL Table Triggers 88
PostgreSQL Table Options 89
PostgreSQL Views 90
PostgreSQL Functions 91
PostgreSQL Aggregates 93
PostgreSQL Conversions 94
PostgreSQL Domains 95
PostgreSQL Indexes 96
PostgreSQL Operators 97
PostgreSQL Operator Classes 98
PostgreSQL Sequences 100
PostgreSQL Triggers 101
PostgreSQL Trigger Functions 102

PostgreSQL Types 104
PostgreSQL Tablespaces 107
PostgreSQL Casts 107
PostgreSQL Languages 108
SQLite Objects 109
SQLite Tables 109
SQLite Table Fields 109
SQLite Table Indexes 111
SQLite Table Foreign Keys 112
SQLite Table Uniques 113
SQLite Table Checks 114
SQLite Table Triggers 114
SQLite Table Options 115
SQLite Views 116
SQLite Indexes 116
SQLite Triggers 117
SQL Server Objects 118
SQL Server Backup/Restore (Available only in Full Version) 123
SQL Server Backup 123
SQL Server Restore 126
SQL Server Schemas 127
SQL Server Tables 127
SQL Server Table Fields 128
SQL Server Table Indexes 129
SQL Server Table Foreign Keys 130
SQL Server Table Uniques 131
SQL Server Table Checks 131
SQL Server Table Triggers 132
SQL Server Table Storage 133
SQL Server Table Options 134
SQL Server Views 135
SQL Server Functions/Procedures 136
SQL Server Indexes 137
SQL Server Synonyms 142
SQL Server Triggers 143
SQL Server Backup Devices 145
SQL Server Linked Servers 145
SQL Server Server Triggers 148
SQL Server Assemblies 149
SQL Server Database Triggers 150
SQL Server Partition Functions 151
SQL Server Partition Schemes 152
SQL Preview 152

Maintain 152
Maintain MySQL/MariaDB 152
Maintain Oracle 153
Maintain PostgreSQL 157
Maintain SQLite 158
Maintain SQL Server 159
Table Viewer 160
Grid View 161
Using Navigation Bar 161
Editing Records 162
Sorting/Finding/Replacing Records 166
Filtering Records (Available only in Full Version) 168
Manipulating Raw Data 168
Formatting Table Grid 168
Form View (Available only in Full Version) 170
Assistant Editors 170
Filter Wizard (Available only in Full Version) 170
Query 173
Query Builder (Available only in Full Version) 173
Query Editor 175
Editor Advanced Features 175
Query Results 178
Query Parameters 179
Debugging Oracle Query (Available only in Full Version) 179
Model (Available only in Navicat Premium and Enterprise Version) 180
Sidebar Panel 180
Model Tree Palette 180
Diagram Tree Palette 181
Properties Palette 181
History Palette 183
Diagram Editor 183
Create Tables 183
Create Notes and Labels 184
Create Images 185
Create Layers 186
Create Relations 187
Format Diagram 188
Preview and Print Model 189
Reverse Engineering 189
Script Generation 189
General Settings for Export SQL 190
Advanced Settings for Export SQL 190
Forward Engineering 191

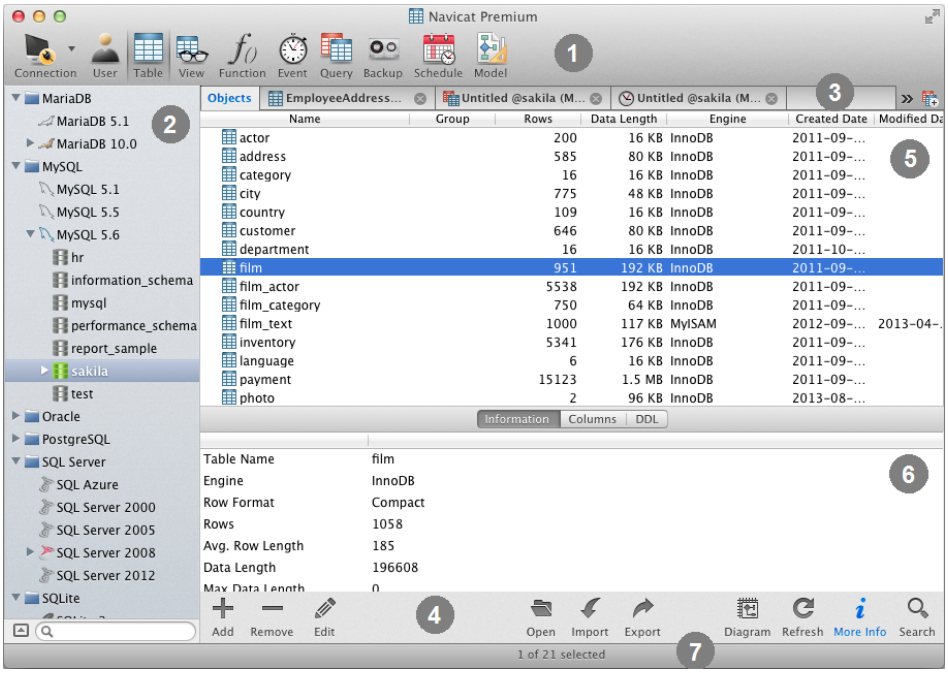
Selecting Synchronization Type 191
Selecting Target Connection 191
Selecting Schemas/Tables 191
Selecting Synchronize Options 191
Viewing Comparison Result 192
Model Hints and Tips 193
Advanced Tools 195
Import Wizard 195
Setting Import File Format 195
Setting Source File Name 195
Setting Delimiter 196
Setting Additional Options 196
Setting Target Table 197
Adjusting Field Structures and Mapping Fields 197
Selecting Import Mode 198
Saving and Confirming Import 199
Export Wizard 199
Setting Export File Format 199
Setting Destination File Name 200
Selecting Fields for Export 200
Setting Additional Options 200
Saving and Confirming Export 201
Data Transfer (Available only in Full Version) 201
General Settings for Data Transfer 201
Advanced Settings for Same Server Type Data Transfer 202
Advanced Settings for Cross Server Data Transfer (Available only in Navicat Premium) 204
Data Synchronization (Available only in Full Version) 206
General Settings for Data Synchronization 206
Advanced Settings for Data Synchronization 207
Structure Synchronization (Available only in Full Version) 207
General Settings for Structure Synchronization 207
Structure Synchronization Result 209
Backup/Restore (Available only in Full Version) 210
Backup 210
Restore 211
Extract SQL 211
Batch Job/Schedule (Available only in Full Version) 211
General Settings for Batch Job/Schedule 213
Advanced Settings for Batch Job/Schedule 213
Batch Job Converter (Available only in Navicat Premium) 214
Dump SQL File/Execute SQL File 214
View Database/Schema/Table Structure (Available only in Full Version) 215
Console 215

Server Security 216
MySQL/MariaDB Security 216
MySQL/MariaDB User Designer 216
Oracle Security 218
Oracle User Designer 218
Oracle Role Designer 219
PostgreSQL Security 220
PostgreSQL Server 7.3 to 8.0 221
PostgreSQL User Designer 221
PostgreSQL Group Designer 222
PostgreSQL Server 8.1 to 9.2 223
PostgreSQL Role Designer 223
SQL Server Security 224
SQL Server Login Designer 225
SQL Server Server Role Designer 228
SQL Server Database User Designer 228
SQL Server Database Role Designer 230
SQL Server Application Role Designer 231
Privilege Manager 232
Useful Tools 233
ER Diagram (Available only in Full Version) 233
Object Information 234
Server Monitor (Available only in Full Version) 235
Virtual Grouping (Available only in Full Version) 236
Connection Colorings 236
Favorites (Available only in Full Version) 236
Find in Database/Schema (Available only in Full Version) 237
Search Filter 237
Preferences 239
General 239
Tabs 239
Grids 240
Fonts and Colors 241
File Paths 241
SQL Editors 242
Models (Available only in Full Version) 243
Environments 243
Commands (Available only in Full Version) 246
Hot Keys 250
Log Files 253

Getting Started

Navicat is a multi-connections Database Administration tool allowing you to connect to MySQL, Oracle, PostgreSQL, SQLite, SQL Server and/or MariaDB databases, making database administration to multiple kinds of database so easy. Features in Navicat are sophisticated enough to provide professional developers for all their specific needs, yet easy to learn for users who are new to database server. With its well-designed Graphical User Interface(GUI), Navicat lets you quickly and easily create, organize, access and share information in a secure and easy way.

Navicat is available on three platforms - Microsoft Windows, Mac OS X and Linux. It can connect users to local/remote server, providing several utility tools such as Data Modeling, Data Transfer, Data/Structure Synchronization, Import/Export, Backup/Restore and Schedule to facilitate the process for data maintenance. For details, visit our web-site: http://www.navicat.com



1 Navicat Main Toolbar

Navicat Main Toolbar allows you to access basic objects and features, such as connections, users, tables, backup, schedule and more.

2 Connection Tree

Connection Tree is the basic way to navigate with connections, databases and database objects. It employs tree structure which allows you to take action upon the database and their objects through their popup menus quickly and easily. To view or hide Connection Tree, choose **View** -> **Show Connection Tree** from main menu.

3 Tab Bar

Tab Bar allows you to switch among Object List and the tabbed windows. You can also choose to always display pop-ups on a new tab, or to always display them in a new window. If you have multiple tabs open, you can use CTRL-TAB to easily switch to other tabs. See also Preferences.

4 Object List Toolbar

Object List Toolbar provides other controls that you can use to manipulate the objects.

Object List

Object List displays a list of objects, such as tables, views, queries and so on.

Object Information

Object Information shows the detailed information of the server objects and Navicat objects. To view or hide the Object Information, click **More Info** from the object list toolbar.

Status Bar

Status Bar displays the current window's status information. To view or hide Status Bar, choose **View** -> **Show Status Bar** from main menu.

System Requirements

System Requirements for Windows

- Microsoft Windows XP SP2, Vista, Windows 7, Windows 8, Server 2003, Server 2008, Server 2012
- 32-bit or 64-bit systems

System Requirements for Mac OS X

- Mac OS X 10.6 Snow Leopard, 10.7 Lion, 10.8 Mountain Lion or 10.9 Mavericks
- Intel CPU

System Requirements for Linux

- Compatible with i386 PC
- Supports 32-bit and 64-bit Linux platform
- Supports Linux kernel version 2.2 or higher
- Supports Glibc 2.4 or above
- Supports GNOME and KDE

Note: You need to install all 32-bit libraries before working on 64-bit Linux.

Registration

To make it economic and efficient for you to purchase our products, over 95% of customers order Navicat via our Online Shop using major Credit Cards - MasterCard, Visa, Euro card, JCB and American Express. All Online orders are processed by **PayPal** and **WorldPay**. The VeriSign Certificate for SSL transactions provided will ensure you a secured Online transactions.

If you have ordered Navicat software and would like to review your order information, or if you have questions about ordering, payments, or shipping procedures, please contact our Navicat Sales Department.

After purchase you will obtain a **Registration Key** to activate your licensed Navicat by e-mail within 24 hours after we received your order. Please make sure to enter a valid e-mail address in your order. If you have not received the keys within 24 hours, it is probably that the e-mail we sent was blocked by your email spam filter. To resend your download information and keys, please

submit your registered email address to our Customer Center. If you get no reply from the resend form, please contact our Navicat Sales Department.

Besides, if you feel uncomfortable with providing your personal information over the Internet, we accept Purchase Order and Bank/Wire Transfer. Please visit our Offline Order.

Installation

We strongly suggest that you shut down any opened applications. This will help ensure a smooth installation.

Note: Installing Navicat does not include the server installation. You should download and install the server manually.
For user who has been trying our unregistered version, just simply key in the **Registration Key** (16 digit) on the pop up Registration screen.

Installation for Download Version

1. Download Navicat Mac OS X version.
2. Open the **.dmg** file.
3. Drag Navicat to your Applications folder to install.
4. After installed, key in the **Registration Key** (16 digit) on the pop up Registration screen and click **Activate** to online activate the key.

Installation for CD Version

1. Load the Navicat CD Installation disk into the CD-ROM drive.
2. Open the **.dmg** file.
3. Drag Navicat to your Applications folder to install.
4. After installed, key in the **Registration Key** (16 digit) on the pop up Registration screen and click **Activate** to online activate the key.

Migrate Navicat to new computer

1. In Navicat, control-click anywhere in the connection tree and choose **Export Connections**. The exported file (.ncx) contains all your connection settings.
2. Backup the exported file (.ncx).
3. In Navicat, choose **Navicat XXX** -> **Registration** and click **Deactivate** to online deactivate the key.
4. Uninstall Navicat from the existing computer.
5. Re-install Navicat in the new computer.
6. Control-click anywhere in the connection tree and choose **Import Connections** in the new computer.

When a new connection is being established, Navicat will create a subfolder, with name in line with the database, under the Settings Location. Most files are stored within this subfolder. To look for the path, control-click the connection and choose **Connection Properties** -> **Advanced** -> **Settings Location**.

Maintenance/Upgrade

How to purchase the maintenance plan?

Navicat Software Maintenance Plan allows Navicat users to receive priority email support, receiving software upgrades and receiving bug fix releases at no additional cost during the protected period.

Subscription to the Maintenance Plan is done at the time of your software license purchase or within 90 days as of your purchase date - it cannot be added to a previously purchased product at a later date. For details, please click here.

How to upgrade your Navicat?

If you want to upgrade installed copy of Navicat to the latest release, please choose **Navicat XXX** -> **Check for Updates** to start the Updater. It will automatically check your installed version. If there is a new version, simply follow the steps in the Updater to upgrade your Navicat. It will replace your previous Navicat and your current settings will remain unchanged.

Or, you can submit your registered email address on the Customer Center to download the latest version installer.

End-User License Agreement

IMPORTANT: THIS SOFTWARE END USER LICENSE AGREEMENT ("EULA") IS A LEGAL AGREEMENT BETWEEN YOU (EITHER AN INDIVIDUAL OR, IF PURCHASED OR OTHERWISE ACQUIRED BY OR FOR AN ENTITY, AN ENTITY) AND PREMIUMSOFT CYBERTECH LTD..READ IT CAREFULLY BEFORE COMPLETING THE INSTALLATION PROCESS AND USING THE SOFTWARE. IT PROVIDES A LICENSE TO USE THE SOFTWARE AND CONTAINS WARRANTY INFORMATION AND LIABILITY DISCLAIMERS. BY INSTALLING AND USING THE SOFTWARE, YOU ARE CONFIRMING YOUR ACCEPTANCE OF THE SOFTWARE AND AGREEING TO BECOME BOUND BY THE TERMS OF THIS AGREEMENT. IF YOU DO NOT AGREE TO BE BOUND BY THESE TERMS, THEN DO NOT INSTALL THE SOFTWARE AND RETURN THE SOFTWARE TO YOUR PLACE OF PURCHASE. THIS EULA SHALL APPLY ONLY TO THE SOFTWARE SUPPLIED BY PREMIUMSOFT CYBERTECH LTD. HEREWITH REGARDLESS OF WHETHER OTHER SOFTWARE IS REFERRED TO OR DESCRIBED HEREIN.

1. Definitions

a. "Non-commercial Version" means a version of the Software, so identified, for use by i) the individual who is a natural person and not a corporation, company, partnership or association or other entity or organization (ii) the individual who is a student, faculty or staff member at an educational institution, and (iii) staff of a non-profit organization or charity organization only. For purposes of this definition, "educational institution" means a public or private school, college, university and other post secondary educational establishment. A non-profit organization is an organization whose primary objective is to support an issue or matter of private interest or public concern for non-commercial purposes.
b. "Not For Resale (NFR) Version" means a version, so identified, of the Software to be used to review and evaluate the Software, only.
c. "PremiumSoft" means PREMIUMSOFT CYBERTECH LTD. and its licensors, if any.
d. "Software" means only the PremiumSoft software program(s) and third party software programs, in each case, supplied by PremiumSoft herewith, and corresponding documentation, associated media, printed materials, and online or electronic documentation.

e. "Unregistered version", "Trial version" or "Demo version" means an unregistered copy of the SOFTWARE ("UNREGISTERED SOFTWARE") which may be used by the USER for evaluation purposes for a period of thirty (30) days following the initial installation of the UNREGISTERED SOFTWARE. At the end of the trial period ("TRIAL PERIOD"), the USER must either register the SOFTWARE or remove it from his system. The UNREGISTERED SOFTWARE may be freely copied and distributed to other users for their evaluation.

f. "Navicat Essentials" means a version of the Software, so identified, to be used for commercial purpose.

2. License Grants

The licenses granted in this Section 2 are subject to the terms and conditions set forth in this EULA:

a. Subject to Section 2(b), you may install and use the Software on a single computer; OR install and store the Software on a storage device, such as a network server, used only to install the Software on your other computers over an internal network, provided you have a license for each separate computer on which the Software is installed and run. Except as otherwise provided in Section 2(b), a license for the Software may not be shared, installed or used concurrently on different computers.

b. In addition to the single copy of the Software permitted in Section 2(a), the primary user of the computer on which the Software is installed may make a second copy of the Software and install it on either a portable computer or a computer located at his or her home for his or her exclusive use, provided that:

    A. the second copy of the Software on the portable or home computer (i) is not used at the same time as the copy of the Software on the primary computer and (ii) is used by the primary user solely as allowed for such version or edition (such as for educational use only),

    B. the second copy of the Software is not installed or used after the time such user is no longer the primary user of the primary computer on which the Software is installed.

c. In the event the Software is distributed along with other PremiumSoft software products as part of a suite of products (collectively, the "Studio"), the license of the Studio is licensed as a single product and none of the products in the Studio, including the Software, may be separated for installation or use on more than one computer.

d. You may make one copy of the Software in machine-readable form solely for backup purposes. You must reproduce on any such copy all copyright notices and any other proprietary legends on the original copy of the Software. You may not sell or transfer any copy of the Software made for backup purposes.

e. You agree that PremiumSoft may audit your use of the Software for compliance with these terms at any time, upon reasonable notice. In the event that such audit reveals any use of the Software by you other than in full compliance with the terms of this Agreement, you shall reimburse PremiumSoft for all reasonable expenses related to such audit in addition to any other liabilities you may incur as a result of such non-compliance.

f. Your license rights under this EULA are non-exclusive.

3. License Restrictions

a. Other than as set forth in Section 2, you may not make or distribute copies of the Software, or electronically transfer the Software from one computer to another or over a network.

b. You may not alter, merge, modify, adapt or translate the Software, or decompile, reverse engineer, disassemble, or otherwise reduce the Software to a human-perceivable form.

c. Unless otherwise provided herein, you may not rent, lease, or sublicense the Software.

d. Other than with respect to a Trial / Demo Version, Non-commercial Lite Version or a Not For Resale Version of the Software, you may permanently transfer all of your rights under this EULA only as part of a sale or transfer, provided you retain no copies, you transfer all of the Software (including all component parts, the media and printed materials, any upgrades, this EULA, the serial numbers, and, if applicable, all other software products provided together with the Software), and the recipient agrees to the terms of this EULA. If the Software is an upgrade, any transfer must include all prior versions of the Software from which you are upgrading. If the copy of the Software is licensed as part of the whole Studio (as defined above), the Software shall be transferred only with and as part of the sale or transfer of the whole Studio, and not separately. You may retain no copies of the Software. You may not sell or transfer any Trial / Demo Version, Non-commercial Lite Version or Not For Resale Version of the Software.

e. Unless otherwise provided herein, you may not modify the Software or create derivative works based upon the Software.

f. Non-commercial Versions of the Software may not be used for, or distributed to any party for, any commercial purpose.

g. Unless otherwise provided herein, you shall not

   A. in the aggregate, install or use more than one copy of the Trial / Demo Version and Non-commercial Lite Version of the Software,

   B. download the Trial / Demo Version and Non-commercial Lite Version of the Software under more than one username,

   C. alter the contents of a hard drive or computer system to enable the use of the Trial / Demo Version of the Software for an aggregate period in excess of the trial period for one license to such Trial / Demo Version,

   D. disclose the results of software performance benchmarks obtained using the Trial / Demo Version or Non-commercial Lite Version to any third party without PremiumSoft prior written consent, or

   E. use the Trial / Demo Version of the Software for a purpose other than the sole purpose of determining whether to purchase a license to a commercial or education version of the software; provided, however, notwithstanding the foregoing, you are strictly prohibited from installing or using the Trial / Demo Version or Non-commercial Lite Version of the Software for any commercial training purpose.

h. You may only use the Not for Resale Version of the Software to review and evaluate the Software.

i. You may receive the Software in more than one medium but you shall only install or use one medium. Regardless of the number of media you receive, you may use only the medium that is appropriate for the server or computer on which the Software is to be installed.

j. You may receive the Software in more than one platform but you shall only install or use one platform.

k. You shall not use the Software to develop any application having the same primary function as the Software.

l. In the event that you fail to comply with this EULA, PremiumSoft may terminate the license and you must destroy all copies of the Software (with all other rights of both parties and all other provisions of this EULA surviving any such termination).

m. This program may include Oracle Instant Client (OCI). You agree that you shall

1. not use of the Oracle Instant Client to the business operations;

2. not assign, give, or transfer the Oracle Instant Client or an interest in them to another individual or entity;

   a. make the Programs available in any manner to any third party for use in the third party's business operations; and

   b. title to the Programs from passing to the end user or any other party;

3. not reverse engineer, disassemble or decompilation the Oracle Instant Client and duplicate the Programs except for a sufficient number of copies of each Program for your licensed use and one copy of each Program media;

4. discontinue use and destroy or return to all copies of the Oracle Instant Client and documentation after termination of the Agreement;

5. not pubish any results of benchmark tests run on the Programs;

6. comply fully with all relevant export laws and regulations of the United States and other applicable export and import laws to assure that neither the Oracle Instant Client, nor any direct product thereof, are exported, directly or indirectly, in violation of applicable laws;
7. allow PremiumSoft to audit your use of the Oracle Instant Client;

4. Upgrades

If this copy of the Software is an upgrade from an earlier version of the Software, it is provided to you on a license exchange basis. You agree by your installation and use of such copy of the Software to voluntarily terminate your earlier EULA and that you will not continue to use the earlier version of the Software or transfer it to another person or entity unless such transfer is pursuant to Section 3.

5. Ownership

The foregoing license gives you limited license to use the Software. PremiumSoft and its suppliers retain all rights, title and interest, including all copyright and intellectual property rights, in and to, the Software (as an independent work and as an underlying work serving as a basis for any application you may develop), and all copies thereof. All rights not specifically granted in this EULA, including Federal and International Copyrights, are reserved by PremiumSoft and its suppliers.

6. LIMITED WARRANTY AND DISCLAIMER

a. Except with respect to Trial / Demo Version, Non-commercial Lite Version and Not For Resale Version of the Software, PremiumSoft warrants that, for a period of thirty (30) days from the date of delivery (as evidenced by a copy of your receipt): the physical media on which the Software is furnished will be free from defects in materials and workmanship under normal use. The Software is provided "as is". PremiumSoft makes no warranties, express or implied, arising from course of dealing or usage of trade, or statutory, as to any matter whatsoever.
b. PremiumSoft provides no remedies or warranties, whether express or implied, for Trial / Demo version, Non-commercial Lite version and the Not for Resale version of the Software. Trial / Demo version, Non-commercial Lite version and the Not for Resale version of the Software are provided "as is".
c. Except as set Forth in the foregoing limited warranty with respect to software other than Trial/ Demo version, Non-commercial Lite version and Not for Resale version, PremiumSoft and its suppliers disclaim all other warranties and representations, whether express, implied, or otherwise, including the warranties of merchantability or fitness for a particular purpose. Also, there is no warranty of non-infringement and title or quiet enjoyment. PremiumSoft does not warrant that the Software is error-free or will operate without interruption. The Software is not designed, intended or licensed for use in hazardous environments requiring fail-safe controls, including without limitation, the design, construction, maintenance or operation of nuclear facilities, aircraft navigation or communication systems, air traffic control, and life support or weapons systems. PremiumSoft specifically disclaims any express or implied warranty of fitness for such purposes.
d. If applicable law requires any warranties with respect to the Software, all such warranties are limited in duration to thirty (30) days from the date of delivery.
e. No oral or written information or advice given by PremiumSoft, its dealers, distributors, agents or employees shall create a warranty or in any way increase the scope of ANY warranty PROVIDED HEREIN.

7. LIMITATION OF LIABILITY

(a) Neither PremiumSoft nor its suppliers shall be liable to you or any third party for any indirect, special, incidental, punitive or consequential damages (including, but not limited to, damages for the inability to use equipment or access data, loss of business, loss of profits, business interruption or the like), arising out of the use of, or inability to use, the Software and based on any theory of liability including breach of contract, breach of warranty,
tort (including negligence), product liability or otherwise, even if PremiumSoft or its representatives have been advised of the possibility of such damages.

8. Third Party Software

The Software may contain third party software which requires notices and/or additional terms and conditions. By accepting this EULA, you are also accepting the additional terms and conditions of the third party software.

9. General

No PremiumSoft dealer, agent or employee is authorized to make any amendment to this EULA.

This EULA contains the complete agreement between the parties with respect to the subject matter hereof, and supersedes all prior or contemporaneous agreements or understandings, whether oral or written. You agree that any varying or additional terms contained in any purchase order or other written notification or document issued by you in relation to the Software licensed hereunder shall be of no effect. The failure or delay of PremiumSoft to exercise any of its rights under this EULA or upon any breach of this EULA shall not be deemed a waiver of those rights or of the breach.

If any provision of this EULA shall be held by a court of competent jurisdiction to be contrary to law, that provision will be enforced to the maximum extent permissible, and the remaining provisions of this EULA will remain in full force and effect.

10. Basis of Bargain

The Limited Warranty and Disclaimer and Limited Liability set forth above are fundamental elements of the basis of the agreement between PremiumSoft and you. PremiumSoft would not be able to provide the Software on an economic basis without such limitations. Such Limited Warranty and Disclaimer and Limited Liability inure to the benefit of PremiumSoft's licensors.

11. Term

By downloading and/or installing this SOFTWARE, the Licensor agrees to the terms of this EULA.

This license is effective until terminated. Licensor has the right to terminate your License immediately if you fail to comply with any term of this License.

"as is". Licensor makes no warranties, express or implied, arising from course of dealing or usage of trade, or statutory, as to any matter whatsoever. In particular, any and all warranties or merchantability, fitness for a particular purpose or non-infringement of third party rights are expressly excluded.

12. Governing Law

This License will be governed by the laws in force in Hong Kong. You hereby consent to the non-exclusive jurisdiction and venue sitting in Hong Kong to resolve any disputes arising under this EULA.

Should you have any questions concerning the validity of this License, please contact: licensing@navicat.com. If you desire to contact the Licensor for any other reason, please contact support@navicat.com.

PremiumSoft and other trademarks contained in the Software are trademarks or registered trademarks of PremiumSoft CyberTech Ltd. in the United States and/or other countries. Third party trademarks, trade names, product names and logos may be the trademarks or registered trademarks of their respective owners. You may not remove or alter any trademark, trade names, product names, logo, copyright or other proprietary notices, legends, symbols or labels in the Software. This EULA does not authorize you to use PremiumSoft or its licensors names or any of their respective trademarks.

Connection

To start working with your server in Navicat, you should first establish a connection or several connections using the connection window. If you are new to the server or 'Net in general' and are not quite sure how things work, you may want to look at:

- MySQL User Manual
- Oracle Database Documentation
- PostgreSQL User Manual
- SQLite User Manual
- SQL Server MSDN Library
- MariaDB Documentation

To create a new connection, click or choose **Connection** -> **New Connection**. Then, enter the necessary information in Connection Properties window.

After you have created your connections, your databases/schemas appear in the connection tree on the left. If the **Show tables in connections tree** option is checked at the Preferences, all database/schema objects are also displayed in the tree. To connect to a database/schema, simply double-click it in the tree.

Note: Navicat authorizes you to make connection to remote servers running on different platforms, i.e. Windows, Mac, Linux and UNIX.

You can edit the connection information by control-click the connection and choose **Connection Properties**. To open a connection settings location, control-click the connection in the connection tree and choose **Open Settings Location**.

Flush MySQL/MariaDB Connection

Flush has several variant forms that clear or reload various internal caches, flush tables, or acquire locks. To execute Flush, you must have the *Reload* privilege, see MySQL/MariaDB Security.

Control-click the connection and select **Flush** from the popup menu.

Privileges	Reload the privileges from the grant tables in the *mysql* database.
Hosts	Empty the host cache tables. You should flush the host tables if some of your hosts change IP number or if you get the error message *Host 'host_name' is blocked*. When more than *max_connect_errors* errors occur in a row for a given host while connection to MySQL server, MySQL assumes something is wrong and blocks the host from further connection requests. Flushing the host tables allow the host to attempt to connect again.
Logs	Close and reopens all log files. If you have specified the update log file or a binary log file without an extension, the extension number of the log file will be incremented by one relative to the previous file. If you have used an extension in the file name, MySQL will close and reopen the update log file.
Status	Reset most status variables to zero. This is something one should only use when debugging a query.
Tables	Close all open tables and forces all tables in use to be closed.

SQL Azure Firewall Settings

You cannot connect to SQL Azure until you have granted your client IP access. To access SQL Azure database from your computer, ensure that your firewall allows outgoing TCP communication on TCP port 1433. You must have at least one firewall rule before you can connection to SQL Azure.

Control-click the SQL Azure connection and select **SQL Azure Firewall Rules** from the popup menu. You can add new rule by providing a range of IP address.

Testing Account

Navicat provides evaluated accounts for testing purpose.

The remote MySQL server connection settings are:

- Host Name/IP Address: server1.navicat.com
- Port: 4406
- User Name: navicat
- Password: testnavicat

The remote PostgreSQL server connection settings are:

- Host Name/IP Address: server1.navicat.com
- Port: 5432
- Initial Database: HR
- User Name: navicat
- Password: testnavicat

General Settings

To successfully establish a new connection to local/remote server - no matter via SSL, SSH or HTTP, set the connection properties in the General tab. If your Internet Service Provider (ISP) does not provide direct access to its server, Secure Tunneling Protocol (SSH) / HTTP is another solution. Enter a friendly name to best describe your connection in **Connection Name** text box.

MySQL and MariaDB connections

You can connect to your MySQL server remotely however for security reasons native remote direct connections to the MySQL server are disabled. Therefore, you cannot use Navicat Premium or other similar MySQL admin applications running on your computer to connect to the remote server directly unless the User Privileges has been configured.

By default, MySQL gives "root" as username and leave the password field blank.

Host Name/IP Address

A host name where the database is situated or the IP address of the server.

Port

A TCP/IP port for connecting to the database server.

User Name

User name for connecting to the database server.

Password

Password for connecting to the server.

Encoding

Choose a codepage to communicate with MySQL Server while MySQL character set not being employed.

Use MySQL character set

This option should be enabled if employing MySQL 4.1 or above.

Use compression

This option allows you to use compression protocol. It is used if both client and server support zlib compression, and the client requests compression.

Oracle connection

Navicat supports 2 types of Oracle server connection. In **Basic** mode, Navicat connects to Oracle through the Oracle Call Interface (OCI). OCI is an application programming interface that allows an application developer to use a third-generation language's native procedure or function calls to access the Oracle database server and control all phases of SQL statement execution. OCI is a library of standard database access and retrieval functions in the form of a dynamic-link library.

In **TNS** mode, Navicat connects to Oracle server using an alias entry from a tnsnames.ora file through the Oracle Call Interface (OCI). OCI is an application programming interface that allows an application developer to use a third-generation language's native procedure or function calls to access the Oracle database server and control all phases of SQL statement execution. OCI is a library of standard database access and retrieval functions in the form of a dynamic-link library.

By default, Oracle created a number of user accounts upon installation. Administrative accounts: SYS, SYSTEM, SYSMAN, and DBSNMP. Sample schema accounts: SCOTT, HR, OE, OC, PM, IX and SH.

Type

Basic	**Host Name/IP Address** A host name where the database is situated or the IP address of the server. **Port** A TCP/IP port for connecting to the database server. **Service Name/SID** Set the Service Name/SID which the user connects when making connection. Select the corresponding radio button.
TNS	User needs to provide the **Net Service Name**.

Role

Indicate that the database user is connecting with either the **Default**, **SYSOPER** or **SYSDBA** system privilege.

OS authentication

With this option on, Oracle Database uses OS user login credentials to authenticate database users.

User Name

User name for connecting to the database server.

Password

Password for connecting to the server.

See also:

Environments

PostgreSQL connection

For security reasons native remote direct connections to the PostgreSQL server are disabled. Therefore, you may not be able to use Navicat Premium or other similar PostgreSQL admin applications running on your computer to connect to the remote server. By default, PostgreSQL only allows connections from the local machine using TCP/IP connections. Other machines will not be able to connect unless you modify *listen_addresses* in the *postgresql.conf* file, enable host-based authentication by modifying the *$PGDATA/pg_hba.conf* file, and restart the server. For more information: Client Authentication

By default, PostgreSQL gives "postgres" as username and leave the password field blank.

Host Name/IP Address

A host name where the database is situated or the IP address of the server.

Port

A TCP/IP port for connecting to the database server.

Default Database

The initial database to which user connects when making connection.

User Name

User name for connecting to the database server.

Password

Password for connecting to the server.

SQLite connection

You can choose the **Type** of the SQLite database and specify the file path.

Existing database file

Connect an existing database in the **Database File**.

New SQLite 3

Create a new SQLite 3 database in the **Database File**.

New SQLite 2

Create a new SQLite 2 database in the **Database File**.

Database File

Specify the initial database file. If the HTTP Tunnel is enabled, you need to enter an absolute file path of the database file in your web server.

SQL Server connection

Host Name/IP Address

A host name where the database is situated or the IP address of the server.

Port

A TCP/IP port for connecting to the database server.

Initial Database

Set the initial database to which the user connects when making connection.

Authentication Type

SQL Server uses two ways to validate connections to SQL Server databases: **Basic** and **Windows Authentication**.

Basic	SQL Server Authentication uses login records to validate the connection. Users must provide their server login: **User Name** and **Password**.
Windows Authentication	When a user connects through a Windows user account, SQL Server validates the account name and password using the Windows principal token in the operating system. This means that the user identity is confirmed by Windows. SQL Server does not ask for the password, and does not perform the identity validation. Users need to provide the **Domain**, **User Name** and **Password**.

Advanced Settings

Settings Location

When a new connection is being established, Navicat will create a subfolder under the Settings Location. Most files are stored within this subfolder:

Navicat Objects	Server Types	File Extensions
Query	All	.sql
Query Builder	All	.qbs - stores the layout of tables in Query Builder.
Import Wizard Profile	MySQL	.npi
	Oracle	.nopi
	PostgreSQL	.nppi
	SQLite	.nlpi
	SQL Server	.nspi
	MariaDB	.nmpi

Export Wizard Profile	MySQL	.npe
	Oracle	.nope
	PostgreSQL	.nppe
	SQLite	.nlpe
	SQL Server	.nspe
	MariaDB	.nmpe
Export Query Result Profile	MySQL	.npeq
	Oracle	.nopeq
	PostgreSQL	.nppeq
	SQLite	.nlpeq
	SQL Server	.nspeq
	MariaDB	.nmpeq
Export View Result Profile	MySQL	.npev
	Oracle	.nopev
	PostgreSQL	.nppev
	SQLite	.nlpev
	SQL Server	.nspev
	MariaDB	.nmpev
Export Materialized View Profile	Oracle	.nopem
Backup	MySQL, PostgreSQL, SQLite and MariaDB	.pmb
Backup Profile	MySQL	.npb
	PostgreSQL	.nppb
	SQLite	.nlpb
	MariaDB	.nmpb
ER Diagram File	All	.ned
Data Pump Export Profile	Oracle	.exp

Other files are located in the default folder, e.g. ~/Library/Application Support/PremiumSoft CyberTech/Navicat Premium. You are allowed to change Virtual Grouping and Model File locations under Preferences.

Other Files	Server Types	File Extensions
Model File	All	.ndm (inside DataModels folder)
Data Transfer	All	.xml (inside Data Transfer Profiles folder)
Structure Synchronization	All	.xml (inside Structure Synchronize Profiles folder)
Data Synchronization	All	.xml (inside Data Synchronize Profiles folder)
Schedule	All	.xml (inside Schedules folder)
Virtual Grouping	All	vgroup.xml - stores how the objects are categorized.

See also:

Log Files

Auto Connect

With this option on, Navicat automatically open connection with the registered database at application startup.

MySQL and MariaDB connections

Socket Timeout (sec)

This option allows you to set the socket timeout value for running query.

Keepalive interval (sec)

This option allows you to keep the connection with the server alive by pinging it. You can set the period between pings in the edit field.

Use socket file

With this option on, Navicat uses socket file for localhost connection.

Oracle and SQL Server connections

Keepalive interval (sec)

This option allows you to keep the connection with the server alive by pinging it. You can set the period between pings in the edit field.

PostgreSQL connection

Keepalive interval (sec)

This option allows you to keep the connection with the server alive by pinging it. You can set the period between pings in the edit field.

Use socket file

With this option on, Navicat uses socket file for localhost connection.

SQLite connection

Encrypted

Enable this option and provide **Password** when connecting to an encrypted SQLite database.

Attached Databases

To attach or detach databases in the connection.

Advanced database properties for MySQL, PostgreSQL and MariaDB

Set the advanced database properties, which are not obligatory. To start working with advanced database settings, check the **Use advanced connections**. The detailed description is given below:

To show the selected databases in the connection tree, click the preferable databases in the Databases list box. The check box will show as

To add a hidden database

1. Click **Add DB to List** button.
2. Enter the database name.
3. Select the newly added database in the Databases list box.

To remove a database, select the database in the Databases list box and click **Remove DB from List** button.

Note: The database will be just removed from the Databases list box, it will still exist in the server.

Advanced database properties for SQLite

You can click **Attach Database** button to attach a database file.

Option	Description
Database Name	Enter the database name which displays in Navicat.
Database File	Set the file path for a database.
Encrypted	Enable this option and provide **Password** when connecting to an encrypted SQLite database.

To detach a database, select it from the list and click **Detach Database** button.

SSL Settings

Secure Sockets Layer(SSL) is a protocol for transmitting private documents via the Internet. To get a secure connection, the first thing you need to do is to install OpenSSL Library and download Database Source.

Note: Available only for MySQL, PostgreSQL and MariaDB.
Support from PostgreSQL 8.4 or later.

MySQL and MariaDB connections

To provide authentication details, enable **Use authentication** and fill in the required information:

Client Key File

The SSL key file in PEM format to use for establishing a secure connection.

Client Certificate File

The SSL certificate file in PEM format to use for establishing a secure connection.

CA Certificate File

The path to a file in PEM format that contains a list of trusted SSL certificate authorities.

Specified Cipher

A list of permissible ciphers to use for SSL encryption.

PostgreSQL connection

Choose the **SSL Mode**:

allow	First try a non-SSL connection; if that fails, try an SSL connection.
prefer	First try an SSL connection; if that fails, try a non-SSL connection.
require	Only try an SSL connection.
verify-ca	Only try an SSL connection, and verify that the server certificate is issued by a trusted CA.
verify-full	Only try an SSL connection, verify that the server certificate is issued by a trusted CA and that the server hostname matches that in the certificate.

To provide authentication details, enable **Use authentication** and fill in the required information:

Client Certificate File

The path of the client certificate.

Client Key File

The path of the client private key.

Root Certificate File

The path of the trusted certificate authorities.

Certificate Revocation List File

The file path of the SSL certificate revocation list (CRL).

SSH Settings

Secure SHell (SSH) is a program to log in into another computer over a network, execute commands on a remote server, and move files from one machine to another. It provides strong authentication and secure encrypted communications between two hosts, known as **SSH Port Forwarding (Tunneling)**, over an insecure network. Typically, it is employed as an encrypted version of Telnet.

In a Telnet session, all communications, including username and password, are transmitted in plain-text, allowing anyone to listen-in on your session and steal passwords and other information. Such sessions are also susceptible to session hijacking, where a malicious user takes over your session once you have authenticated. SSH serves to prevent such vulnerabilities and allows you to access a remote server's shell without compromising security.

Note: Available only for MySQL, Oracle, PostgreSQL, SQL Server and MariaDB.

Please make sure that the parameter - "AllowTcpForwarding" in the Linux Server must be set to value "yes", otherwise, the SSH port forwarding will be disabled. To look for the path: /etc/ssh/sshd_config. By default, the SSH port forwarding should be enabled. Please double check the value settings.

** Even the server support SSH tunnel, however, if the port forwarding being disabled, Navicat cannot connect via SSH Port 22.

Host Name/IP Address

A host where SSH server is activated.

Port

A port where SSH server is activated, by default it is 22.

User Name

A user on SSH server machine. (It is not a user of database server.)

Authentication Method

Password	Provide the SSH server user **Password**.
Public Key	**Private Key** It is used together with your public key. The private key should be readable only by you. **Passphrase** A passphrase is exactly like a password, except that it applies to the keys you are generating and not an account.

Use compression

Request compression of all data (including stdin, stdout, stderr, and data for forwarded X11 and TCP connections). The compression algorithm is the same used by gzip(1), and the "level" can be controlled by the CompressionLevel option for protocol version 1.

Note: Navicat host name at the General tab should be set relatively to the SSH server which provided by your database hosting company.

HTTP Settings

HTTP Tunneling is a method for connecting to a server that uses the same protocol (http://) and the same port (port 80) as a web server does. It is used while your ISPs do not allow direct connections, but allows establishing HTTP connections.

Note: Available only for MySQL, PostgreSQL, SQLite and MariaDB.

Uploading the Tunneling Script

To use this connection method, first thing you need to do is to upload the tunneling script to the web server where your server is located.

Note: Click the **Save Tunnel Script As** button to extract the script file, **ntunnel_mysql.php**, **ntunnel_pgsql.php**, **ntunnel_sqlite.php** and **ntunnel_mariadb.php**.

Setting up HTTP Tunnel

The following instruction guides you through the process of configuring a HTTP connection.

1. Click or choose **Connection** -> **New Connection** to set up the Connection Properties.

2. Select the HTTP tab and enable **Use HTTP tunnel**.
3. Enter URL of the tunneling script, e.g. *http://www.navicat.com/ntunnel_mysql.php* .
4. If the tunneling script is hosted in a password protected server, you can provide the required authentication details.
5. If your server installed a Web Application Firewall, you can check the **Encode query as base64** option.
6. Navicat host name at the General Settings page should be set relatively to the HTTP server which provided by your database hosting company.

Note: HTTP Tunnel and SSH Tunnel cannot function simultaneously. The SSH Tunnel is disabled when you select the HTTP Tunnel and vice versa.

Server Objects

Navicat provides powerful tools to manage server objects, such as databases, tables, views, functions, etc.

Note: Before working with the server objects in Navicat, you should establish the connection first.

MySQL/MariaDB Objects

To start working with the server objects, you should create and open a connection. If the server is empty, you need to control-click the connection in the connection tree and choose **New Database** to create a new database.

To edit an existing database properties, control-click the database in the connection tree and choose **Database Properties**. Please notice that MySQL does not support renaming database through its interface at this moment. Access the directory in which databases being stored. By default, all databases store within a directory called **data** under MySQL Installation folder. For example: /usr/local/mysql5/data.

Database Name

Set the name for a new database.

Default Character Set

Specify the default database character set.

Default Collation

Specify the default database collation.

MySQL/MariaDB Tables

Relational databases use tables to store data. All operations on data are done on the tables themselves or produce another table as the result. A table is a set of rows and columns, and their intersections are fields. From a general perspective, columns within a table describe the name and type of data that will be found by row for that column's fields. Rows within a table represent records composed of fields that are described from left to right by their corresponding column's name and type. Each field in a row is implicitly correlated with each other field in that row. Click to open an object list for **Table**.

When open a table with graphical fields, control-click a table and select **Open Table (Quick)** from the popup menu. Faster performance for opening the graphical table, as BLOB fields (images) will not be loaded until you click on the cell. If you do wish Navicat loads all your images while opening the table, click **Open** button from the object list toolbar.

You can create a table shortcut by drag the table out. It provides a convenient way for you to open your table for entering data directly without activating the main Navicat.

To empty a table, control-click the selected table and choose **Empty Table** from the popup menu. This option is only applied when you wish to clear all the existing records without resetting the auto-increment value. To reset the auto-increment value while emptying your table, use **Truncate Table**.

MySQL/MariaDB Table Fields

In the **Fields** tab, just simply click a field for editing. By using the field toolbar, you can create new, insert and drop the selected field. To search a field name, choose **Edit** -> **Find** -> **Find** or press CMD-F.

Note: **Insert Field** button is supported from MySQL 3.22 or later.

You can change the order of a field, simply drag and drop to field to desired location. To enable the drag and drop function, control-click the fields grid and choose **Drag to change table columns' order**.

Note: Support from MySQL 4.0.1 or later.

Use the **Name** edit box to set the field name. Note that the name of the field must be unique among all the field names in the table.

The **Type** dropdown list defines the type of the field data. See MySQL Data Types/MariaDB Data Types for details.

Use the **Length** edit box to define the length of the field and use **Decimals** edit box to define the number of digits after the decimal point (the scale) for Floating Point data type.

Note: Be careful when shortening the field length as losing data might be caused.

Not Null

Check this box to not allow the NULL values for the field.

Virtual

Check this box to enable the virtual column settings for the field.

Note: Support from MariaDB 5.2 or later.

Key

A Primary Key is a single field or combination of fields that uniquely defines a record. None of the fields that are part of the primary key can contain a null value.

Field's Properties

Note: The following options depend on the field type you are chosen.

Default Value

Set the default value for the field.

Comment

Set any optional text describing the current field.

Column Format

Set the column format for the field.

Storage

Set the storage for the field.

Key Length

The edit box will be enabled when Primary Key is set.

Character Set

A character set is a set of symbols and encodings.

Collation

A collation is a set of rules for comparing characters in a character set.

Note: MySQL chooses the column character set and collation in the following manner:

- If both CHARACTER SET X and COLLATE Y were specified, then character set X and collation Y are used.
- If CHARACTER SET X was specified without COLLATE, then character set X and its default collation are used.
- Otherwise, the table character set and collation are used.

Binary

As of MySQL 4.1, values in CHAR and VARCHAR fields are sorted and compared according to the collation of the character set assigned to the field.

Before MySQL 4.1, sorting and comparison are based on the collation of the server character set; you can declare the field with the BINARY attribute to cause sorting and comparison to be based on the numeric values of the bytes in field values. BINARY does not affect how field values are stored or retrieved.

Auto Increment

The AUTO INCREMENT attribute can be used to generate a unique identity for new rows. To start with the AUTO INCREMENT value other than 1, you can set that value in Options tab.

Unsigned

UNSIGNED values can be used when you want to allow only non-negative numbers in a field and you need a bigger upper numeric range for the field.

As of MySQL 4.0.2, floating-point and fixed-point types also can be UNSIGNED. Unlike the integer types, the upper range of column values remains the same.

Zerofill

The default padding of spaces is replaced with zeros. For example, for a field declared as INT(5) ZEROFILL, a value of 4 is retrieved as 00004; for a field declared as FLOAT(20,10) ZEROFILL, a value of 0.1 is retrieved as 000000000.1000000015.

Note: If you specify ZEROFILL for a numeric type, MySQL automatically adds the UNSIGNED attribute to the field.

On Update Current_Timestamp

As of 4.1.2, you have more flexibility in deciding which TIMESTAMP field automatically is initialized and updated to the current timestamp.

Enum Values

Use this edit box to define the members of SET/ENUM.

Type

Choose the virtual column's type. **VIRTUAL** means that the column is calculated on the fly when a command names it. **PERSISTENT** means that the value is physically stored in the table.

Note: Support from MariaDB 5.2 or later.

Expression

Specify an expression or function to evaluate at insertion time.

Note: Support from MariaDB 5.2 or later.

MySQL/MariaDB Table Indexes

Indexes are organized versions of specific columns in your tables. MySQL uses indexes to facilitate quick retrieval of records. With indexes, MySQL can jump directly to the records you want. Without any indexes, MySQL has to read the entire data file to find the correct record(s).

In the **Indexes** tab, just simply click an index field for editing. By using the index toolbar, you can create new, edit and delete the selected index field.

Use the **Name** edit box to set the index name.

To include field(s) in the index, just click the **Fields** to open the editor for editing. Select the field(s) from the list. To remove the fields from the index, uncheck them in the same way. You can also use the arrow buttons to change the index field(s) order. The **Key Length** edit box(s) is used to set index KEY LENGTH.

Note: Some of data types do not allow indexing by several fields. For example: BLOB

Index Type

Define the type of the table index.

Normal	NORMAL indexes are the most basic indexes, and have no restraints such as uniqueness.
Unique	UNIQUE indexes are the same as NORMAL indexes with one difference - all values of the indexed column(s) must only occur once.
Full Text	FULL TEXT indexes are used by MySQL in full-text searches.

Index Method

Specify an index type when creating an index, BTREE or HASH.

Comment

Set any optional text describing the current index.

Note: Support from MySQL 5.5.3 or later.

MySQL/MariaDB Table Foreign Keys

A foreign key is a field in a relational table that matches the primary key column of another table. The foreign key can be used to cross-reference tables.

In the **Foreign Keys** tab, just simply click a foreign key field for editing. By using the foreign key toolbar, you can create new, edit and delete the selected foreign key field.

Both tables must be *InnoDB* type (or *solidDB* type if you have solidDB for MySQL). In the referencing table, there must be an index where the foreign key columns are listed as the first columns in the same order. Starting with MySQL 4.1.2, such an index will be created on the referencing table automatically if it does not exist.

Note: Foreign Key support from MySQL 3.23.44 or later.
Editing foreign key is supported from MySQL 4.0.13 or later.
Delete Foreign Key is supported from MySQL 4.0.13 or later.

Use the **Name** edit box to enter a name for the new key.

Use the **Referenced Database** and **Referenced Table** dropdown lists to select a foreign database and table respectively.

To include field(s)/referenced field(s) to the key, click **Fields** or **Referenced Fields** to open the editor(s) for editing.

The **On Delete** and **On Update** dropdown list define the type of the actions to be taken.

CASCADE	Delete the corresponding foreign key, or update the corresponding foreign key to the new value of the primary key.
SET NULL	Set all the columns of the corresponding foreign key to NULL.
No ACTION	Does not change the foreign key.
RESTRICT	Produce an error indicating that the deletion or update would create a foreign key constraint violation.

Related topic:

Foreign Keys Data Selection

MySQL/MariaDB Table Triggers

A trigger is a named database object that is associated with a table and that is activated when a particular event occurs for the table.

In the **Triggers** tab, just simply click a trigger field for editing. By using the trigger toolbar, you can create new, edit and delete the selected trigger field.

Note: Trigger is supported from MySQL 5.0.2 or later.

Use the **Name** edit box to set the trigger name.

Use the **Fires** dropdown list to define the trigger action time. It can be **Before** or **After** to indicate that the trigger activates before or after the statement that activated it.

Insert

The trigger is activated whenever a new row is inserted into the table. For example, **INSERT**, **LOAD DATA**, and **REPLACE** statements.

Update

The trigger is activated whenever a row is modified. For example, **UPDATE** statement.

Delete

The trigger is activated whenever a row is deleted from the table. For example, **DELETE** and **REPLACE** statement. However, **DROP TABLE** and **TRUNCATE** statements on the table do not activate the trigger.

The **Statement** edit box defines the statement to execute when the trigger activates. To include your statement, just simply click to write. If you want to execute multiple statements, use the **BEGIN ... END** compound statement construct. Example:

```
BEGIN
  set new.capacity = new.capacity + 100;
  set new.amount = new.amount + 100;
END
```

MySQL/MariaDB Table Options

Table Type

Define the engine of the table. Use the **Set Default** button to set the default table type.

Auto Increment

Set/Reset the **Auto Increment** value in the edit field. The auto increment value indicates the value for next record.

Default Character Set

Define the type of the character set for table.

Default Collation

Choose the collation for the table.

Data Directory

To specify where the MyISAM storage engine should put a table's data file.

Index Directory

To specify where the MyISAM storage engine should put a table's index file.

Max Rows

The maximum number of rows you plan to store in the table. This is not a hard limit, but rather a hint to the storage engine that the table must be able to store at least this many rows.

Average Row Length

An approximation of the average row length for your table. You need to set this only for large tables with variable-size rows.

Min Rows

The minimum number of rows you plan to store in the table.

Key Block Size

This option provides a hint to the storage engine about the size in bytes to use for index key blocks. The engine is allowed to change the value if necessary. A value of 0 indicates that the default value should be used.

Note: Support from MySQL 5.5 or later.

Row Format

Define how the rows should be stored.

Pack Keys *(take effect only with MyISAM table)*

Set this option to 1 if you want to have smaller indexes. This usually makes updates slower and reads faster. Setting the option to 0 disables all packing of keys. Setting it to **DEFAULT** tells the storage engine to pack only long *CHAR*, *VARCHAR*, *BINARY*, or *VARBINARY* columns.

Checksum *(only for MyISAM table)*

Check this option if you want MySQL to maintain a live checksum for all rows.

Delay key write *(only for MyISAM table)*

Check this option if you want to delay key updates for the table until the table is closed.

Page Checksum *(only for Aria table)*

Check this option if you want index and data use page checksums for extra safety.

Note: Support from MariaDB 5.1 or later.

Transactional *(only for Aria table)*

Check this option if you want crash-safe.

Note: Support from MariaDB 5.1 or later.

Union Tables *(only for MRG_MYISAM table)*

UNION is used when you want to access a collection of identical *MyISAM* tables as one. This works only with *MERGE* tables. You must have *SELECT*, *UPDATE*, and *DELETE* privileges for the tables you map to a *MERGE* table.

Insert Method *(only for MRG_MYISAM table)*

If you want to insert data into a *MERGE* table, you must specify with INSERT_METHOD the table into which the row should be

inserted. INSERT_METHOD is an option useful for *MERGE* tables only. Use a value of **FIRST** or **LAST** to have inserts go to the first or last table, or a value of **NO** to prevent inserts.

Connection *(only for FEDERATED table)*

To create the local table that will be federated to the remote table. You can create the local table and specify the connection string (containing the server name, login, password) to be used to connect to the remote table using the **Connection** edit box.

The CONNECTION string contains the information required to connect to the remote server containing the table that will be used to physically store the data. The connection string specifies the server name, login credentials, port number and database/table information.

The format the connection string is as follows:

scheme://user_name[:password]@host_name[:port_num]/db_name/tbl_name

Sample of connection strings:

CONNECTION='mysql://username:password@hostname:port/database/tablename'

CONNECTION='mysql://username@hostname/database/tablename'

CONNECTION='mysql://username:password@hostname/database/tablename'

Tablespace *(only for ndbcluster table)*

To specify the tablespace for the storage.

Note: Support from MySQL 5.1.6 or later.

Storage *(only for ndbcluster table)*

To specify type of storage used (disk or memory), and can be one of **DISK**, **MEMORY**, or **DEFAULT**.

Note: Support from MySQL 5.1.6 or later.

Partition Options

Set the Partition Options.

Note: Support from MySQL 5.1 or later.

Option	Description
Partition by	Select the function that is used to determine the partition: **Hash**, **Key**, **Range** and **List**.
Partitions no.	Set the partition number.
Subpartition by	Select the function that is used to determine the subpartition: **Hash** and **Key**.
Subpartitions no.	Set the subpartition number.
Partition Definition	Use **Add Partition** or **Remove Partition** to add or delete the partition. Use **Add Subpartition** or **Remove Subpartition** to add or delete the subpartition.
Values	For range partitioning, each partition must include a VALUES LESS THAN clause; for list partitioning, you must specify a VALUES IN clause for each partition. This is used to determine which rows are to be stored in this partition.

Engine	Select the storage engine for both partition and subpartition.
Data Directory	The directory where the data for this partition are to be stored.
Index Directory	The directory where the indexes for this partition are to be stored.
Max Rows	The maximum number of rows to be stored in the partition.
Min Rows	The minimum number of rows to be stored in the partition.
Tablespace	Designate a tablespace for the partition. Used for Falcon only.
Nodegroup	Set the Node Group.
Comment	Enter the comment for the partition.

MySQL/MariaDB Views

Views (including updatable views) are implemented in MySQL Server 5.0 and available in binary releases from 5.0.1 and up. Views are useful for allowing users to access a set of relations (tables) as if it were a single table, and limiting their access to just that. Views can also be used to restrict access to rows (a subset of a particular table). For access control to columns, you can also use the sophisticated privilege system in MySQL Server. Click to open an object list for **View**.

You can create a view shortcut by drag the view out. It provides a convenient way for you to open your view without activating the main Navicat.

Button	Description
Preview	Preview and/or Explain the view.

Note: You can choose to show the Result tab below the editor or in a new tab by selecting **Edit** -> **Show Result** -> **Below Query Editor** or **In a New Tab**.

View Builder (Available only in Full Version)

View Builder allows you to build views visually. It allows you to create and edit views without knowledge of SQL. See Query Builder for details.

View Editor

You can edit the view definition as SQL statement (SELECT statement it implements).

Hint: To customize the view of the editor and find out more features for sql editing, see Editor Advanced Features.

Options

View's Algorithm

Undefined	MySQL chooses which algorithm to use. It prefers Merge over Temptable if possible, because Merge is usually more efficient and because a view cannot be updatable if a temporary table is used.
Merge	The text of a statement that refers to the view and the view definition are merged such that parts of the view

	definition replace corresponding parts of the statement.
Temptable	The results from the view are retrieved into a temporary table, which then is used to execute the statement.

With check option

Local	Restrict the Check option only to the view being defined.
Cascaded	Cause the checks for underlying views to be evaluated as well.

SQL Security

The SQL SECURITY characteristic determines which MySQL account to use when checking access privileges for the view when the view is executed. The legal characteristic values are **Definer** and **Invoker**. These indicate that the view must be executable by the user who defined it or invoked it, respectively.

Definer

The default Definer value is the user who executes the *CREATE VIEW* statement. (This is the same as DEFINER = CURRENT_USER.) If a user value is given, it should be a MySQL account in 'user_name'@'host_name' format (the same format used in the *GRANT* statement). The user_name and host_name values both are required.

View Viewer

View Viewer displays the view data as a grid. Data can be displayed in two modes: **Grid View** and **Form View**. See Table Viewer for details.

MySQL/MariaDB Functions/Procedures

Stored routines (procedures and functions) are supported in MySQL 5.0. A stored routine is a set of SQL statements that can be stored in the server. Once this has been done, clients do not need to keep reissuing the individual statements but can refer to the stored routine instead. Click to open an object list for **Function**.

Definition

Definition consists of a valid SQL procedure statement. This can be a simple statement such as *SELECT* or *INSERT*, or it can be a compound statement written using *BEGIN* and *END*. Compound statements can contain declarations, loops, and other control structure statements.

Hint: To customize the view of the editor and find out more features for sql editing, see Editor Advanced Features.

Parameter

Define function/procedure parameter.

Return Type

This text box will be enabled only for creating a function. It indicates the return type of the function.

Type

Select the stored routines you wish to create from the drop-down list, i.e. **PROCEDURE** and **FUNCTION**.

Advanced Properties

SQL Security

Specify whether the routine should be executed using the permissions of the user who creates the routine or the user who invokes it.

Definer

The default Definer value is the user who executes the *CREATE PROCEDURE* or *CREATE FUNCTION* statement. (This is the same as DEFINER = CURRENT_USER.) If a user value is given, it should be a MySQL account in 'user_name'@'host_name' format (the same format used in the *GRANT* statement). The user_name and host_name values both are required.

SQL Data Access

Several characteristics provide information about the nature of data use by the routine.

CONTAINS SQL	Indicate that the routine does not contain statements that read or write data. It is the default if none of these characteristics is given explicitly.
NO SQL	Indicate that the routine contains no SQL statements.
READS SQL DATA	Indicate that the routine contains statements that read data, but not statements that write data.
MODIFIES SQL DATA	Indicate that the routine contains statements that may write data.

Deterministic

A procedure or function is considered deterministic if it always produces the same result for the same input parameters, and not deterministic otherwise.

Result

To run the procedure/function, click **Execute** on the toolbar. If the SQL statement is correct, the statement will be executed and, if the statement is supposed to return data, the **Result** tab opens with the data returned by the procedure/function. If an error occurs while executing the procedure/function, execution stops, the appropriate error message is displayed. If the function/procedure requires input parameter, the **Input Parameters** box will popup.

MySQL/MariaDB Events

MySQL Event Scheduler was added in MySQL 5.1.6. MySQL Events are tasks that run according to a schedule. Therefore, we sometimes refer to them as scheduled events. When you create an event, you are creating a named database object containing one or more SQL statements to be executed at one or more regular intervals, beginning and ending at a specific date and time. Conceptually, this is similar to the idea of the Windows Task Scheduler. Click to open an object list for **Event**.

Definition

Definition consists of a valid SQL statement. This can be a simple statement such as *SELECT* or *INSERT*, or it can be a compound statement written using *BEGIN* and *END*. Compound statements can contain declarations, loops, and other control structure statements.

Hint: To customize the view of the editor and find out more features for sql editing, see Editor Advanced Features.

Definer

Specify the user account to be used when checking access privileges at event execution time. The default DEFINER value is the user who executes the *CREATE EVENT* statement. (This is the same as DEFINER = CURRENT_USER.) If a user value is given, it should be a MySQL account in 'user_name'@'host_name' format (the same format used in the GRANT statement). The user_name and host_name values both are required.

Status

You can create an event but keep it from being active using the *DISABLE* keyword. Alternatively, you may use *ENABLE* to make explicit the default status, which is active.

On Completion

Normally, once an event has expired, it is immediately dropped. You can override this behavior by specifying *ON COMPLETION PRESERVE*. Using *ON COMPLETION NOT PRESERVE* merely makes the default non-persistent behavior explicit.

Schedule

At

AT timestamp is used for a one-time event. It specifies that the event executes one time only at the date and time, given as the *timestamp*, which must include both the date and time, or must be an expression that resolves to a datetime value. Use **INTERVAL** to create an event which occurs at some point in the future relative to the current date and time.

Every

For actions which are to be repeated at a regular interval, you can use an *EVERY* clause which followed by an *interval*. (**INTERVAL** is not used with *EVERY*.)

Starts

An *EVERY* clause may also contain an optional *STARTS* clause. *STARTS* is followed by a *timestamp* value which indicates when the action should begin repeating, and may also use **INTERVAL** interval in order to specify an amount of time "from now".

For example: ***EVERY 3 MONTH STARTS CURRENT_TIMESTAMP + 1 WEEK*** means "every three months, beginning one week from now".

Ends

An *EVERY* clause may also contain an optional *ENDS* clause. The *ENDS* keyword is followed by a *timestamp* value which tells MySQL when the event should stop repeating. You may also use **INTERVAL** interval with *ENDS*.

For example: ***EVERY 12 HOUR STARTS CURRENT_TIMESTAMP + INTERVAL 30 MINUTE ENDS CURRENT_TIMESTAMP + INTERVAL 4 WEEK*** is equivalent to "every twelve hours, beginning thirty minutes from now, and ending four weeks from now".

P.S. The *timestamp* must be in the future - you cannot schedule an event to take place in the past.

The *interval* portion consists of two parts, a quantity and a *unit of time.

*YEAR | QUARTER | MONTH | DAY | HOUR | MINUTE |

WEEK | SECOND | YEAR_MONTH | DAY_HOUR | DAY_MINUTE |

DAY_SECOND | HOUR_MINUTE | HOUR_SECOND | MINUTE_SECOND

Oracle Objects

To start working with the server objects, you should create and open a connection. When you create a user account, you are also implicitly creating a schema for that user. A schema is a logical container for the database objects (such as tables, views, triggers, and so on) that the user creates. The schema name is the same as the user name, and can be used to unambiguously refer to objects owned by the user. Other user schemas are showed under **Other Schemas**.

Hint: Oracle interprets non-quoted object identifiers as uppercase. In Navicat, all objects identifier will be quoted. That is, Navicat saves exactly what you have inputted.

Oracle Data Pump (Available only in Full Version)

Oracle Data Pump technology enables very high-speed movement of data and metadata from one database to another. It includes two utilities: Data Pump Export and Data Pump Import.

Data Pump Export is a utility for unloading data and metadata into a set of operating system files called a dump file set. The dump file set can be imported only by the Data Pump Import utility. The dump file set can be imported on the same system or it can be moved to another system and loaded there.

Data Pump Import is a utility for loading an export dump file set into a target system. The dump file set is made up of one or more disk files that contain table data, database object metadata, and control information. The files are written in a proprietary, binary format. During an import operation, the Data Pump Import utility uses these files to locate each database object in the dump file set.

Click to open an object list for **Oracle Data Pump**.

You can change the Directory of the dump file set by control-click anywhere in the object list and select **Change Directory** from the popup menu.

Note: Support from Oracle 10g or later.
Data Pump requires SYSDBA Role and the dump file set is stored in the server.

Oracle Data Pump Export

You can save the Data Pump Export settings to a profile. Simply click the **Save** button.

Hint: The profiles(.exp) are saved under the Settings Location.

To show the hidden tabs (advanced options), check the **Show advanced options** box.

General Properties

Job Name

The name of the job.

Mode

FULL	In a full database export, the entire database is unloaded. This mode requires that you have the EXP_FULL_DATABASE role.
TABLESPACE	In tablespace mode, only the tables contained in a specified set of tablespaces are unloaded. If a table is unloaded, its dependent objects are also unloaded. Both object metadata and data are unloaded.
SCHEMAS	If you have the EXP_FULL_DATABASE role, then you can specify a list of schemas and optionally include the schema definitions themselves, as well as system privilege grants to those schemas. If you do not have the EXP_FULL_DATABASE role, you can export only your own schema.
TABLE	In table mode, only a specified set of tables, partitions, and their dependent objects are unloaded. You must have the EXP_FULL_DATABASE role to specify tables that are not in your own schema. All specified tables must reside in a single schema.

Content

ALL	Unload both data and metadata.
DATA_ONLY	Unload only table row data; no database object definitions are unloaded.
METADATA_ONLY	Unload only database object definitions; no table row data is unloaded.

Output Files

Add dump files to the dump file set for the export.

Objects to be exported

Select the objects to export. If you select TABLE mode, choose the schema in the **Schema** dropdown list.

Metadata Filter

Include or **Exclude** a set of objects from the Export operation. Choose the **Object Type** and specify the **Name Clause**.

Data Filter

Enable Table Data Filters

Specify a subquery that is added to the end of the SELECT statement for the table.

Enable Sample Filters

Specify a percentage for sampling the data blocks to be moved.

Remap Data

Table Schema

The schema containing the column to be remapped.

Table Name

The table containing the column to be remapped.

Column Name

The name of the column to be remapped.

Package Schema

The schema of the package.

Package Name

The name of the package.

Package Function

A PL/SQL package function which is called to modify the data for the specified column.

Encryption

Encryption Content

Specify what to encrypt in the dump file set.

ALL	Enable encryption for all data and metadata in the export operation.
DATA_ONLY	Only data is written to the dump file set in encrypted format.
ENCRYPTED_COLUMNS_ONLY	Only encrypted columns are written to the dump file set in encrypted format.
METADATA_ONLY	Only metadata is written to the dump file set in encrypted format.
NONE	No data is written to the dump file set in encrypted format.

Encryption Algorithm

Identify which cryptographic algorithm should be used to perform encryption.

Encryption Mode

Option	Description
Transparent	Allow an encrypted dump file set to be created without any intervention from a database administrator (DBA), provided the required Oracle Encryption Wallet is available.
Encryption Password	Provide a password when creating encrypted dump file sets.
Dual	Create a dump file set that can later be imported using either the Oracle Encryption Wallet or the password that was specified with the ENCRYPTION_PASSWORD parameter.

Encryption Password

Specify a key for re-encrypting encrypted table columns, metadata, or table data so that they are not written as clear text in the dump file set.

Confirm Password

Re-type your password.

Advanced Properties

Number of worker processes to be used

The maximum number of worker processes that can be used for the job.

Reuse files

A preexisting file will be overwritten.

Append timestamp to dump and log file names

Check this box to append timestamp to dump and log file names.

Enable data options for XMLCLOBS

Check this box to enable data options for XMLCLOBS.

Version

The version of database objects to be extracted.

COMPATIBLE	The version of the metadata corresponds to the database compatibility level and the compatibility release level for feature.
LATEST	The version of the metadata corresponds to the database version.

Database Link

The name of a database link to the remote database that will be the source of data and metadata for the current job.

Estimate

Specify that the estimate method for the size of the tables should be performed before starting the job.

Compression Type

ALL	Compress both user data and metadata.
DATA_ONLY	Compress only user data in the dump file set.
METADATA_ONLY	Compress only metadata in the dump file set.
NONE	Store the dump file set in an uncompressed format.

Transportable

Operate on metadata for tables (and their dependent objects) within a set of selected tablespaces to perform a transportable tablespace export.

Directory

Choose the log file directory.

Log File Name

Enter the name of the log file.

Flashback SCN

System change number (SCN) to serve as transactionally consistent point for reading user data.

Flashback Time

Either the date and time used to determine a consistent point for reading user data or a string of the form TO_TIMESTAMP(...).

Oracle Data Pump Import

To show the hidden tabs (advanced options), check the **Show advanced options** box.

General Properties

Job Name

The name of the job.

Mode

FULL	In a full database export, the entire database is unloaded. This mode requires that you have the EXP_FULL_DATABASE role.
TABLESPACE	In tablespace mode, only the tables contained in a specified set of tablespaces are unloaded. If a table is unloaded, its dependent objects are also unloaded. Both object metadata and data are unloaded.
SCHEMAS	If you have the EXP_FULL_DATABASE role, then you can specify a list of schemas and optionally include the schema definitions themselves, as well as system privilege grants to those schemas. If you do not have the EXP_FULL_DATABASE role, you can export only your own schema.
TABLE	In table mode, only a specified set of tables, partitions, and their dependent objects are unloaded. You must have the EXP_FULL_DATABASE role to specify tables that are not in your own schema. All specified tables must reside in a single schema.

Content

ALL	Unload both data and metadata.
DATA_ONLY	Unload only table row data; no database object definitions are unloaded.
METADATA_ONLY	Unload only database object definitions; no table row data is unloaded.

Input Files

Add dump files to the dump file set for the import.

Objects to be imported

Select the objects to import. If you select TABLE mode, specify the schema in the **Schema** text box.

Network

Database Link

The name of a database link to the remote database that will be the source of data and metadata for the current job.

Estimate

Specify that the estimate method for the size of the tables should be performed before starting the job.

Flashback SCN

System change number (SCN) to serve as transactionally consistent point for reading user data.

Flashback Time

Either the date and time used to determine a consistent point for reading user data or a string of the form TO_TIMESTAMP(...).

Transportable

Operate on metadata for tables (and their dependent objects) within a set of selected tablespaces to perform a transportable tablespace export.

DataFile Path

Specify the full file specification for a datafile in the transportable tablespace set.

Filters

Enable Include/Exclude Filters

Include or **Exclude** a set of objects from the Import operation. Choose the **Object Type** and specify the **Name Clause**.

Enable Table Data Filters

Specify a subquery that is added to the end of the SELECT statement for the table. If you specify a WHERE clause in the subquery, you can restrict the rows that are selected.

Remap Data

Enable Data Filters

Fields	Description
Table Schema	The schema containing the column to be remapped.
Table name	The table containing the column to be remapped.
Column Name	The name of the column to be remapped.
Package Schema	The schema of the package.
Package Name	The name of the package.
Package Function	A PL/SQL package function which is called to modify the data for the specified column.

Enable DataFiles Filters

Specify a remapping to be applied to objects as they are processed in the specified job. Enter the **Source DataFile** and **Target DataFile**.

Remap Objects

Enable Remap Schema

Specify a remapping to be applied to schemas as they are processed in the specified job. Enter the **Source schema** and choose the **Target schema**.

Enable Remap TableSpace

Specify a remapping to be applied to tablespaces as they are processed in the specified job. Enter the **Source TableSpace** and choose the **Target TableSpace**.

Enable Remap Table

Specify a remapping to be applied to tables as they are processed in the specified job. Enter the **Source Table** and choose the **Target Table**.

Advanced Properties

Number of worker processes to be used

The maximum number of worker processes that can be used for the job.

Reuse data files

Check this box to reuse existing datafiles for tablespace creation.

Skip unusable indexes

Check this box to skip loading tables that have indexes that were set to the Index Unusable state (by either the system or the user).

Streams Configuration

Check this box to import any general Streams metadata that may be present in the export dump file.

Action on table if table exists

Specify the action to be performed when data is loaded into a preexisting table.

SKIP	The preexisting table is left unchanged.
APPEND	New rows are added to the existing rows in the table.
TRUNCATE	Rows are removed from a preexisting table before inserting rows from the Import.
REPLACE	Preexisting tables are replaced with new definitions. Before creating the new table, the old table is dropped.

Data Options

A bitmask to supply special options for processing the job.

Partition Options

Specify how partitioned tables should be handled during an import operation.

NONE	Partitioning is reproduced on the target database as it existed in the source database.
DEPARTITION	Each partition or subpartition that contains storage in the job is reproduced as a separate unpartitioned table.
MERGE	Each partitioned table is re-created in the target database as an unpartitioned table.

Version

The version of database objects to be extracted.

COMPATIBLE	The version of the metadata corresponds to the database compatibility level and the compatibility release level for feature.
LATEST	The version of the metadata corresponds to the database version.

Encryption Password

Specify a key for re-encrypting encrypted table columns, metadata, or table data so that they are not written as clear text in the dump file set.

Segment Attributes

Designate the segment attribute to which the transform applies.

Storage

Designate the storage to which the transform applies.

OID

Designate the OID to which the transform applies.

PCTSpace

Specify a percentage multiplier used to alter extent allocations and datafile sizes. Used to shrink large tablespaces for testing purposes.

Directory

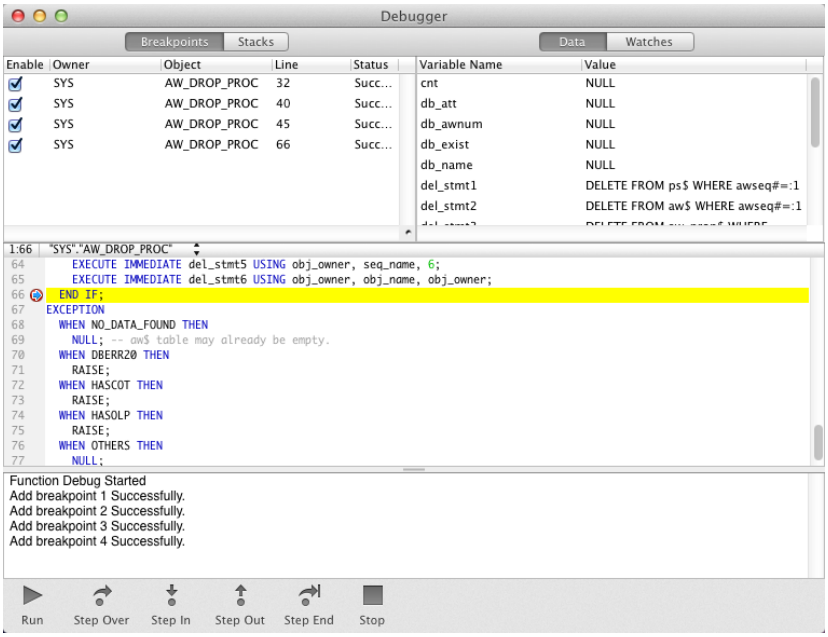
Choose the log file directory.

Log File Name

Enter the name of the log file.

Oracle Debugger (Available only in Full Version)

Navicat provides Oracle PL/SQL debugger for debugging Oracle functions, procedures, packages and queries.



You can perform the most commonly used actions for debugging on the toolbar or menu:

Button	Description
Run	Start running code in debug mode. The debugger executes your code until the end of the code or the next breakpoint is reached. Keyboard shortcut: OPTION-CMD-ENTER
Step Over	Resume the execution. The current line will be executed. If the line is a procedure or function call, it will bypass the procedure or function. The counter will then move to the next line of code. Keyboard shortcut: SHIFT-CMD-O
Step In	Resume the execution. The current line will be executed. If the line is a procedure or function call, the counter goes to the first statement in the procedure or function. Otherwise, the counter will move to the next line of code. Keyboard shortcut: SHIFT-CMD-I
Step Out	Resume the execution. The remaining part of the code within the current procedure or function will be executed.
Step End	Resume the execution. The counter will jump to the last line of the procedure or function. Keyboard shortcut: SHIFT-CMD-E
Stop	Stop stepping the code. The execution will stop and cannot resume it. Keyboard shortcut: SHIFT-CMD-ENTER

The **Breakpoints** tab displays all the breakpoints which allowing you to delete, enable or disable breakpoints. To enable/disable a breakpoint, simply check/uncheck the **Enable** box. Also, you can delete a breakpoint or enable/disable all breakpoints, simply control-click a breakpoint and choose **Delete**, **Enable All** or **Disable All**.

The **Stacks** pane displays the procedure or function calls of the current line.

The **Data** pane displays information about the variables associated with breakpoints.

The **Watches** tab displays information about the variables being watched, allowing you to add, delete or edit watch variables. To add a watch variable, simply control-click anywhere of the Watches tab and choose **Add**. Then, enter the **Variable Name** and **Value** of the variable. To delete a watch variable, simply control-click a variable and choose **Delete**.

The **Code Window** shows the code of the procedure or function, etc. You can add/remove breakpoints for debugging by clicking in the grey area beside each statement. To add a variable to the watch list, simply control-click the highlighted code and choose **Add to Watch List**. To show the debug tips, simply mouse-over the code.

The **Log** pane shows the message log and output when debugging the code.

Oracle Physical Attributes/Default Storage Characteristics

Pct Free

Specify a whole number representing the percentage of space in each data block of the database object reserved for future updates to rows of the object.

Pct Used

Specify a whole number representing the minimum percentage of used space that Oracle maintains for each data block of the database object. A block becomes a candidate for row insertion when its used space falls below this value.

Ini Trans

Specify the initial number of concurrent transaction entries allocated within each data block allocated to the database object.

Max Trans

Specify the maximum number of concurrent update transactions allowed for each data block in the segment.

Initial

Specify the size of the first extent of the object. Use the dropdown list K or M to specify the size in kilobytes or megabytes.

Next

Specify the size of the next extent to be allocated to the object. Use the dropdown list K or M to specify the size in kilobytes or megabytes.

Min

Specify the total number of extents to allocate when the object is created.

Max

Specify the total number of extents, including the first, that Oracle can allocate for the object. Check **Unlimited** if you want extents to be allocated automatically as needed.

Pct Increase

Specify the percent by which the third and subsequent extents grow over the preceding extent.

Buffer Pool

KEEP	Choose this to put blocks from the segment into the KEEP buffer pool. Maintaining an appropriately sized KEEP buffer pool lets Oracle retain the schema object in memory to avoid I/O operations. KEEP takes precedence over any NOCACHE clause you specify for a table, cluster, materialized view, or materialized view log.

RECYCLE	Choose this to put blocks from the segment into the RECYCLE pool. An appropriately sized RECYCLE pool reduces the number of objects whose default pool is the RECYCLE pool from taking up unnecessary cache space.

Free Lists

For objects other than tablespaces and rollback segments, specify the number of free lists for each of the free list groups for the table, partition, cluster, or index.

Free List Groups

Specify the number of groups of free lists for the database object you are creating.

Optimal

Specify an optimal size for a rollback segment. Use the dropdown list K, M, G, T, P or E to specify the size in kilobytes, megabytes, gigabytes, terabytes, petabytes, or exabytes. Check **Null** for no optimal size for the rollback segment.

Oracle Tables

Relational databases use tables to store data. All operations on data are done on the tables themselves or produce another table as the result. A table is a set of rows and columns, and their intersections are fields. From a general perspective, columns within a table describe the name and type of data that will be found by row for that column's fields. Rows within a table represent records composed of fields that are described from left to right by their corresponding column's name and type. Each field in a row is implicitly correlated with each other field in that row. Click to open an object list for **Table**.

To create a new normal table, simply click **Add** from the object list toolbar. Or, you can click-and-hold on **Add** to choose the type **Normal** / **External** / **Index Organized**.

When open a table with graphical fields, control-click a table and select **Open Table (Quick)** from the popup menu. Faster performance for opening the graphical table, as BLOB fields (images) will not be loaded until you click on the cell. If you do wish Navicat loads all your images while opening the table, click **Open** button from the object list toolbar.

You can create a table shortcut by drag the table out. It provides a convenient way for you to open your table for entering data directly without activating the main Navicat.

To empty a table, control-click the selected table and choose **Empty Table** from the popup menu. This option is only applied when you wish to clear all the existing records without resetting the auto-increment value. To reset the auto-increment value while emptying your table, use **Truncate Table**.

Oracle Normal Tables

Tables are the basic unit of data storage in an Oracle database. Data is stored in rows and columns. You define a table with a table name and set of columns.

In a normal (heap-organized) table, data is stored as an unordered collection (heap).

Oracle Table Fields

In the **Fields** tab, just simply click a field for editing. By using the field toolbar, you can create new and drop the selected field. To search a field name, choose **Edit** -> **Find** -> **Find** or press CMD-F.

Use the **Name** edit box to set the field name. Note that the name of the field must be unique among all the field names in the table.

The **Type** dropdown list defines the type of the field data. See Oracle Built-in Datatypes for details.

Use the **Size** edit box to define the **precision** (total number of digits) of the field and use **Scale** edit box to define the **scale** (number of digits to the right of the decimal point) for **numeric** column.

Note: Be careful when shortening the field length as it might result in data loss.

Not Null

Check this box to not allow the NULL values for the field.

Key

A Primary Key is a single field or combination of fields that uniquely defines a record. None of the fields that are part of the primary key can contain a null value.

Field's Properties

Note: The following options depend on the field type you are chosen.

Default Value

Set the default value for the field.

Comment

Set any optional text describing the current field.

Day Precision

Set the number of digits in the leading field.

Fractional Seconds Precision

Set the number of digits in the fractional part of the SECOND datetime field.

Year Precision

Set the number of digits in the year.

Unit

Set the unit either in BYTE or CHAR.

Schema

Set the schema for the field type.

User-defined Type

Set the type for the field.

Oracle Table Indexes

Indexes are optional structures associated with tables and clusters. You can create indexes on one or more columns of a table to speed SQL statement execution on that table. An Oracle Database index provides a faster access path to table data. Indexes are the primary means of reducing disk I/O when properly used.

You can create many indexes for a table as long as the combination of columns differs for each index. You can create more than one index using the same columns if you specify distinctly different combinations of the columns.

In the **Indexes** tab, just simply click an index field for editing. By using the index toolbar, you can create new, edit and delete the selected index field.

Use the **Name** edit box to set the index name.

To include field(s) in the index, click **Fields** to open the editor for editing.

Index Type

Define the type of the table index.

Non-unique	Non-unique indexes do not impose the restriction of unique indexes on the column values.
Unique	Unique indexes guarantee that no two rows of a table have duplicate values in the key column (or columns).
Bitmap	In a bitmap index, a bitmap for each key value is used instead of a list of rowids.

Parallel with degree

Parallel indexing can improve index performance when you have a large amount of data, and have multiple CPUs. Enter the degree that determines the number of separate indexing processes.

Schema

The schema in which to create the index.

Note: To create an index in your own schema, at least one of the following conditions must be true:

- The table or cluster to be indexed is in your own schema.
- You have INDEX privilege on the table to be indexed.
- You have CREATE ANY INDEX system privilege.

To create an index in another schema, all of the following conditions must be true:

- You have CREATE ANY INDEX system privilege.
- The owner of the other schema has a quota for the tablespaces to contain the index or index partitions, or UNLIMITED TABLESPACE system privilege.

Oracle Table Foreign Keys

A foreign key specifies that the values in a column (or a group of columns) must match the values appearing in some row of another table. We say this maintains the referential integrity between two related tables.

In the **Foreign Keys** tab, just simply click a foreign key field for editing. By using the foreign key toolbar, you can create new, edit and delete the selected foreign key field.

Use the **Name** edit box to enter a name for the new key.

Use the **Referenced Schema**, **Referenced Table** and **Referenced Constraint** dropdown lists to select a foreign schema, table and constraint respectively.

To include field(s) to the key, click **Local Fields** to open the editor(s) for editing.

The **On Delete** dropdown list defines the type of the actions to be taken.

No Action	This is the default action. Referenced key values will not be updated or deleted.
Cascade	Delete any rows referencing the deleted row, or update the value of the referencing column to the new value of the referenced column, respectively.
Set Null	Set the referencing column(s) to null.

Enable

You can choose whether to enable/disable the foreign key constraint by checking/unchecking the box.

Related topic:

Foreign Keys Data Selection

Oracle Table Uniques

Unique constraints ensure that the data contained in a column or a group of columns is unique with respect to all the rows in the table.

In the **Uniques** tab, just simply click an unique field for editing. By using the unique toolbar, you can create new, edit and delete the selected unique field.

Use the **Name** edit box to set the unique name.

To set field(s) as unique, click **Fields** to open the editor(s) for editing.

Enable

You can choose whether to enable/disable the unique constraint by checking/unchecking the box.

Oracle Table Checks

A check constraint is the most generic constraint type. It allows you to specify that the value in a certain column must satisfy a Boolean (truth-value) expression.

In the **Checks** tab, just simply click a check field for editing. By using the check toolbar, you can create new, edit and delete the selected check field.

Use the **Name** edit box to set the check name.

Expression

Type in the definition for the check constraint. That is, set the condition for checking, e.g. "field_name1 > 0 AND field_name2 > field_name1" in the **Expression** edit box. A check constraint specified as a column constraint should reference that column's value only, while an expression appearing in a table constraint may reference multiple columns.

Enable

You can choose whether to enable/disable the check constraint by checking/unchecking the box.

Oracle Table Triggers

A trigger is a specification that the database should automatically execute a particular function whenever a certain type of operation is performed. Triggers can be defined to execute either before or after any INSERT, UPDATE, or DELETE operation, either once per modified row, or once per SQL statement.

In the **Triggers** tab, just simply click a trigger field for editing. By using the trigger toolbar, you can create new, edit and delete the selected trigger field.

Name

Set the trigger name.

Compound

Check to set the trigger as a compound trigger.

Note: Support from Orac1e 11g or later.

For Each

Set the trigger as a row trigger or statement trigger.

Fires

Specify the trigger timing whether the trigger action is to be run before or after the triggering statement.

Insert

Fire the trigger whenever an INSERT statement adds a row to a table or adds an element to a nested table.

Update

Fire the trigger whenever an UPDATE statement changes a value in one of the columns specified in **Update Of**. If no **Update Of** are present, the trigger will be fired whenever an UPDATE statement changes a value in any column of the table or nested table.

Delete

Fire the trigger whenever a DELETE statement removes a row from the table or removes an element from a nested table.

Update Of

Specify the fields for UPDATE statement trigger upon necessary. Click to select field(s).

Enable

You can choose whether to enable / disable the trigger constraint by checking / unchecking the box.

Body

Type in the definition for the trigger. Example:

```
BEGIN
  add_job_history(:old.employee_id, :old.hire_date, sysdate,
    :old.job_id, :old.department_id);
END;
```

Old

Specify correlation names. The default correlation name is OLD.

New

Specify correlation names. The default correlation name is NEW.

When

Specify the trigger condition, which is a SQL condition that must be satisfied for the database to fire the trigger. This condition must contain correlation names and cannot contain a query.

Schema

Define the trigger on the current schema.

Follows

Specify the relative firing order of triggers of the same type.

Note: Support from Orac1e 11g or later.

Oracle Table Options

Tablespace

Define a tablespace different from the default tablespace to create a table.

Logging

Specify whether creation of a database object will be logged in the redo log file (LOGGING) or not (NOLOGGING).

Table Compression

Specify whether to compress data segments to reduce disk use. It is valid only for heap-organized tables.

COMPRESS	Enable table compression.
COMPRESS FOR ALL OPERATIONS	Attempt to compress data during all DML operations on the table.
COMPRESS FOR DIRECT_LOAD OPERATIONS	Attempt to compress data during direct-path INSERT operations when it is productive to do so.
NOCOMPRESS	Disable table compression.

Cache

Indicate how blocks are stored in the buffer cache.

CACHE	Indicate that the blocks retrieved for this table are placed at the most recently used end of the least recently used (LRU) list in the buffer cache when a full table scan is performed.
NOCACHE	Indicate that the blocks retrieved for this table are placed at the least recently used end of the LRU list in the buffer cache when a full table scan is performed.

Parallel with degree

Specify the degree of parallelism, which is the number of parallel threads used in the parallel operation.

Row movement

With the option on, it allows the database to move a table row. It is possible for a row to move, for example, during table compression or an update operation on partitioned data.

Physical Attributes

Refer to Physical Attributes/Default Storage Characteristics.

Oracle External Tables

External tables access data in external sources as if it were in a table in the database. While creating external tables, you are actually creating metadata in the data dictionary that enables you to access external data.

Note that external tables are read only. No DML operations are possible and no index can be created.

Fields for Oracle External Tables

In the **Fields** tab, just simply click a field for editing. By using the field toolbar, you can create new and drop the selected field. To search a field name, choose **Edit** -> **Find** -> **Find** or press CMD-F.

Use the **Name** edit box to set the field name. Note that the name of the field must be unique among all the field names in the table.

The **Type** dropdown list defines the type of the field data. See Oracle Built-in Datatypes for details.

Use the **Size** edit box to define the **precision** (total number of digits) of the field and use **Scale** edit box to define the **scale** (number of digits to the right of the decimal point) for **numeric** column.

Note: Be careful when shortening the field length as it might result in data loss.

Field's Properties

Note: The following options depend on the field type you are chosen.

Day Precision

Set the number of digits in the leading field.

Fractional Seconds Precision

Set the number of digits in the fractional part of the SECOND datetime field.

Year Precision

Set the number of digits in the year.

Unit

Set the unit either in BYTE or CHAR.

Schema

Set the schema for the field type.

User-defined Type

Set the type for the field.

External Properties for Oracle External Tables

Default Directory

Specify the default directory for the external table.

Directory

Set the external directory.

Location

Set the external source location.

Access Driver Type

Specify the access driver for the external table. The default type for external tables is ORACLE_LOADER.

Reject Limit

Specify the limit on the number of errors that can occur during a query of the external data.

Parallel with degree

Check to enable parallel query on the data sources and specify the degree of parallel access.

Access Parameters for Oracle External Tables

Describe the mapping of the external data to the Oracle Database data columns.

Using CLOB subquery

Check this option to get a CLOB data value of the returned query.

Oracle Index Organized Tables

An index-organized table has a storage organization that is a variant of a primary B-tree. Data for an index-organized table is stored in a B-tree index structure in a primary key sorted manner. Each leaf block in the index structure stores both the key and nonkey columns.

Index-organized tables have full table functionality. They support features such as constraints, triggers etc with additional features such as key compression.

Note: The Table Designer for **Index Organized Tables** differs from **Normal Tables** only on the **Options** tab. Therefore, you can refer to the Normal Table on the similar tabs.

Options for Oracle Index Organized Tables

Tablespace

Define a tablespace different from the default tablespace to create a table.

Logging

Specify whether creation of a database object will be logged in the redo log file (LOGGING) or not (NOLOGGING).

Parallel with degree

Specify the degree of parallelism, which is the number of parallel threads used in the parallel operation.

Key compress

Check this option to enable key compression. Upon necessary, you can also specify the prefix length (as the number of key columns), which identifies how the key columns are broken into a prefix and suffix entry.

Pct Threshold

When an overflow segment is being used, it defines the maximum size of the portion of the row that is stored in the index block, as a percentage of block size.

Mapping table

Specify if there is a mapping table for the index-organized table. Note that a mapping table is required for creating bitmap indexes on an index-organized table.

Physical Attributes

Refer to Physical Attributes/Default Storage Characteristics.

IOT Overflow

Option	Description
Overflow	Check to enable an overflow storage area. Note: After saving the table, this option cannot be unchecked.
Last Index Column	Specify the column to be put in a separate overflow data segment.
Tablespace	Specify the tablespace in which the overflow segment to be stored.
Logging	Specify whether creation of a database object will be logged in the redo log file (LOGGING) or not (NOLOGGING).
Physical Attributes	Refer to Physical Attributes/Default Storage Characteristics.

Oracle Views

Views are useful for allowing users to access a set of relations (tables) as if it were a single table, and limiting their access to just that. Views can also be used to restrict access to rows (a subset of a particular table). Click to open an object list for **View**.

You can create a view shortcut by drag the view out. It provides a convenient way for you to open your view without activating the main Navicat.

Button	Description
Preview	Preview and/or Explain the view.

Note: You can choose to show the Result tab below the editor or in a new tab by selecting **Edit** -> **Show Result** -> **Below Query Editor** or **In a New Tab**.

View Builder (Available only in Full Version)

View Builder allows you to build views visually. It allows you to create and edit views without knowledge of SQL. See Query Builder for details.

View Editor

You can edit the view definition as SQL statement (SELECT statement it implements).

Hint: To customize the view of the editor and find out more features for sql editing, see Editor Advanced Features.

Advanced Properties

Read only

Indicate that the table or view cannot be updated.

Check option

Indicate that Oracle Database prohibits any changes to the table or view that would produce rows that are not included in the subquery.

Constraint Name

Specify the name. If you omit this identifier, then Oracle automatically assigns a name of the form SYS_Cn, where n is an integer that makes the constraint name unique within the database.

Force on create

Check this option if you want to create the view regardless of whether the base tables of the view or the referenced object types exist or the owner of the schema containing the view has privileges on them.

View Viewer

View Viewer displays the view data as a grid. Data can be displayed in two modes: **Grid View and Form View**. See Table Viewer for details.

Oracle Functions/Procedures

A procedure or function is a schema object that consists of a set of SQL statements and other PL/SQL constructs, grouped together, stored in the database, and run as a unit to solve a specific problem or perform a set of related tasks. Procedures and functions are identical except that functions always return a single value to the caller, while procedures do not. Click to open an object list for **Function**.

Function Pop-up Window

Click the **Add** from the object list toolbar. A Window will pop-up and it allows you to create a procedure/function easily.

Type

Define whether it is a procedure/function.

Name

Specify the name for the procedure/function.

Define the parameter(s) for the procedure/function. Set the parameter **Name**, **Type**, **Mode** and **Default Value** under corresponding columns.

Definition

The **Code Outline** window displays information about the function/procedure including parameter, code body, etc. To show the **Code Outline** window, simply choose **Edit** -> **Show Code Outline** from main menu.

Note: Available only in Full Version.

Button	Description

	Refresh the code outline.
	Turn mouse over highlight on or off.
	Show the code outline in a detail way.
	Sort by type and name.
	Expand the selected item.
	Collapse the selected item.

Hint: To customize the view of the editor and find out more features for sql editing, see Editor Advanced Features.

Result

To run the procedure/function, click **Execute** on the toolbar. If the SQL statement is correct, the statement will be executed and, if the statement is supposed to return data, the **Result** tab opens with the data returned by the procedure/function. If an error occurs while executing the procedure/function, execution stops, the appropriate error message is displayed. If the function/procedure requires input parameter, the **Input Parameters** box will popup.

Debug (Available only in Full Version)

To debug the function/procedure, click **Debug** on the toolbar to launch the Oracle Debugger. Enter the Input Parameters if necessary.

You can add/remove breakpoints for debugging by clicking in the grey area beside each statement.

Oracle Database Links

Database link is a named schema object that describes a path from one database to another and are implicitly used when a reference is made to a global object name in a distributed database. After you have created a database link, you can use it to refer to tables and views on the other database. Click -> **Database Link** to open an object list for **Database Link**.

General Properties

Service Name

Specify the service name of a remote database.

User Name

The user name used to connect to the remote database using a fixed user database link.

Password

The password for connecting to the remote database.

Current User

With this option checked, a current user database link is created. The current user must be a global user with a valid account on the remote database.

Shared

Fill in **Authentication User Name** and **Authentication Password** when Shared option is enabled.

Oracle Indexes

Index provides a faster access path to table data. It is created using one or more columns of a table to speed SQL statement execution on that table. Click -> **Index** to open an object list for **Index**.

You can choose the index **Type**:

Non-unique	A normal index does not impose restrictions on the column values.
Unique	A unique index indicates that no two rows of a table have duplicate values in the key columns.
Bitmap	A bitmap index created with a bitmap for each distinct key, rather than indexing each row separately. Bitmap indexes store the rowids associated with a key value as a bitmap. Each bit in the bitmap corresponds to a possible rowid.
Domain	A domain index is an index designed for a specialized domain, such as spatial or image processing. Users can build a domain index of a given type after the designer creates the indextype.
Cluster	A cluster index is an index designed for a cluster.

General Properties for Non-unique and Unique Indexes

Table Schema

The schema that contains the index.

Table Name

The table name.

Fields

Use the **Field** dropdown list to select the field name and **Order** dropdown list to define the order of the index (ASC or DESC).

General Properties for Bitmap Index

Table Schema

The schema that contains the index.

Table Name

The table name.

Bitmap Join Index

In addition to a bitmap index on a single table, you can create a bitmap join index, which is a bitmap index for the join of two or more tables. A bitmap join index is a space efficient way of reducing the volume of data that must be joined by performing restrictions in advance.

Fields

Use the **Schema**, **Table** and/or **Field** dropdown lists to select the schema, table and/or field name.

Bitmap Join

Use the **Left Schema**, **Left Table**, **Left Field**, **Right Schema**, **Right Table** and **Right Field** dropdown lists to select joined schemas, tables and fields respectively.

General Properties for Domain Index

Table Schema

The schema that contains the index.

Table Name

The table name.

Column

The column which the index is based.

Schema

The schema of the indextype.

Name

Select the created or built-in indextypes.

Parameters

Information about the path table and about the secondary indexes corresponding to the components of XMLIndex.

General Properties for Cluster Index

Cluster Schema

The schema that contains the index.

Cluster Name

The name of the cluster.

Advanced Properties

Unusable

An unusable index must be rebuilt, or dropped and re-created, before it can be used.

Tablespace

The name of the tablespace to hold the index.

Compress

To enable key compression, which eliminates repeated occurrence of key column values and may substantially reduce storage.

Note: No compression for Bitmap Indexes.

Parallel

The creation of the index will be parallelized.

Reverse

To store the bytes of the index block in reverse order, excluding the rowid.

Logging

Choose **LOGGING** to log the creation of the index in the redo log file. Or, choose **NOLOGGING** for no log.

Visibility

Specify the index is **VISIBLE** or **INVISIBLE** to the optimizer.

Online

To indicate that DML operations on the table will be allowed during creation of the index.

No sort

To indicate to the database that the rows are already stored in the database in ascending order, so that Oracle Database does not have to sort the rows when creating the index.

Physical Attributes

Set the physical attributes of an index.

Oracle Java

Java is an object-oriented programming language efficient for application-level programs. You can write and load applications within the database. Click -> **Java** to open an object list for **Java**.

You can choose the **Type**: Source, Class or Resource.

General Properties for Source

BFile

Select the **Directory** and type the **File Name**.

Load From File

Browse the **File** path of Java source file.

Plain Source

Type the source code in the **Plain Source** box.

General Properties for Class and Resource

BFile

Select the **Directory** and type the **File Name**.

Load From File

Browse the **File** path of Java class or Java resource file.

Advanced Properties

Invoker Rights

Select **Current User** to indicate that the methods of the class execute with the privileges of current user or **Definer** to indicate that the methods of the class execute with the privileges of the owner of the schema in which the class resides, and that external names resolve in the schema where the class resides.

Resolver

Specify a mapping of the fully qualified Java name to a Java schema object.

Compile / Resolve

Check this to specify that Oracle Database should attempt to resolve the Java schema object that is created if this statement succeeds.

No force

Check this to roll back the results of the CREATE command of Java if you have enabled Compile or Resolve and the resolution or compilation fails. If you do not specify this option, then Oracle Database takes no action if the resolution or compilation fails, and the created schema object remains.

Oracle Materialized Views

Materialized view is a schema object that can be used to summarize, compute, replicate, and distribute data. Click -> **Materialized View** to open an object list for **Materialized View**.

Button	Description
Preview	Preview and/or Explain the materialized view.

Note: You can choose to show the Result tab below the editor or in a new tab by selecting **Edit** -> **Show Result** -> **Below Query Editor** or **In a New Tab**.

View Builder (Available only in Full Version)

View Builder allows you to build views visually. It allows you to create and edit views without knowledge of SQL. See Query Builder for details.

View Editor

You can edit the view definition as SQL statement (SELECT statement it implements).

Hint: To customize the view of the editor and find out more features for sql editing, see Editor Advanced Features.

Advanced Properties

When

DEMAND	The materialized view will be refreshed on demand by calling one of the three DBMS_MVIEW refresh procedures.
COMMIT	A fast refresh is to occur whenever the database commits a transaction that operates on a master table of the materialized view.
AUTOMATIC	The database automatically refresh the materialized view with the automatic refresh time.
NEVER	The materialized view will not be refreshed with any Oracle Database refresh mechanism or packaged procedure.

Start With

A datetime expression for the first automatic refresh time.

Next

A datetime expression for calculating the interval between automatic refreshes.

Method

FORCE	When a refresh occurs, Oracle Database will perform a fast refresh if one is possible or a complete refresh if fast refresh is not possible.
FAST	A incremental refresh method, which performs the refresh according to the changes that have occurred to the master tables.
COMPLETE	A complete refresh method, which is implemented by executing the defining query of the materialized view.

Type

PRIMARY KEY	A primary key materialized view.
ROW ID	A rowid materialized view.

Master

The remote rollback segment is used at the remote master site for the individual materialized view.

Local

The remote rollback segment is used for the local refresh group that contains the materialized view.

Constraints

ENFORCED	Oracle Database use enforced constraints during the refresh operation.
TRUSTED	Oracle Database use dimension and constraint information that has been declared trustworthy by the database administrator but that has not been validated by the database.

No Index

Check this to suppress the creation of the default index.

Build Type

IMMEDIATE	The materialized view is to be populated immediately.

DEFERRED	The materialized view is to be populated by the next refresh operation.
PREBUILT	To register an existing table as a preinitialized materialized view.

Prebuilt Option

WITH REDUCED PRECISON	To authorize the loss of precision that will result if the precision of the table or materialized view columns do not exactly match the precision returned by subquery.
WITHOUT REDUCED PRECISON	To require that the precision of the table or materialized view columns match exactly the precision returned by subquery, or the create operation will fail.

Compress

COMPRESS	Data segments are compressed to reduce disk and memory use.
NOCOMPRESS	No data segments are compressed.

Parallel

Choose **NOPARALLEL** for serial execution or **PARALLEL** if you want Oracle to select a degree of parallelism equal to the number of CPUs available on all participating instances times the value of the PARALLEL_THREADS_PER_CPU initialization parameter.

With Degree

Set the default degree of parallelism for queries and DML on the materialized view after creation.

Logging

Choose **LOGGING** for logging the creation of Materialized view in the redo log file. Choose **NOLOGGING** for no logging.

Tablespace

Choose the tablespace in which the materialized view is to be created.

Cache

CACHE	The blocks retrieved for the table are placed at the most recently used end of the least recently used (LRU) list in the buffer cache when a full table scan is performed.
NOCACHE	The blocks are placed at the least recently used end of the LRU list.

For update

Check this to allow a subquery, primary key, object, or rowid materialized view to be updated. When used in conjunction with Advanced Replication, these updates will be propagated to the master.

Enable query rewrite

The materialized view is used for query rewrite.

Physical Attributes

Set the Physical Attributes of the materialized view.

Using Index Clause

Option	Description
Tablespace	Choose the tablespace of the index.
Physical Attributes	Set the Physical Attributes for the default index Oracle Database uses to maintain the materialized view data.

View Viewer

View Viewer displays the view data as a grid. Data can be displayed in two modes: **Grid View** and **Form View**. See Table Viewer for details.

Oracle Materialized View Logs

Materialized view log is a schema object that records changes to a master table's data so that a Materialized View defined on the master table can be refreshed incrementally. Click -> **Materialized View Log** to open an object list for **Materialized View Log**.

General Properties

Master Table

The table of the materialized view log.

Tablespace

The tablespace of the materialized view log.

Logging

To specify either **LOGGING** or **NOLOGGING** to establish the logging characteristics for the materialized view log.

Cache

CACHE	The blocks retrieved for this log are placed at the most recently used end of the least recently used (LRU) list in the buffer cache when a full table scan is performed.
NOCACHE	The blocks are placed at the least recently used end of the LRU list.

New Values

INCLUDING	To save both new and old values in the log.
EXCLUDING	To disable the recording of new values in the log.

Parallel

To determine the number of parallel threads used in the parallel operation.

Physical Attributes

Set the Physical Attributes of a materialized view log.

Object ID

The system-generated or user-defined object identifier of every modified row should be recorded in the materialized view log.

Primary Key

The primary key of all rows changed should be recorded in the materialized view log.

Row ID

The rowid of all rows changed should be recorded in the materialized view log.

Sequence

A sequence value providing additional ordering information should be recorded in the materialized view log.

Filter Columns

Choose the columns whose values you want to be recorded in the materialized view log for all rows that are changed.

Oracle Packages

Packages are encapsulated collections of related procedures, stored functions, and other program objects stored together in the database. Package bodies, specified subsequently, defines these objects. An package consists of two parts: a specification and a body. Click -> **Package** to open an object list for **Package**.

The **Code Outline** window displays information about the package/package body including function, procedure, parameter, code body, etc. To show the **Code Outline** window, simply choose **Edit** -> **Show Code Outline** from main menu.

Note: Available only in Full Version.

Button	Description
	Refresh the code outline.
	Turn mouse over highlight on or off.
	Show the code outline in a detail way.
	Sort by type and name.
	Expand the selected item.
	Collapse the selected item.

Package's Definition

Enter the package's definition. After saving the package, you can edit the Package Body. Just click **New Package Body** or **Edit Package Body** to open the Package Body Designer.

Hint: To customize the view of the editor and find out more features for sql editing, see Editor Advanced Features.

Package Body's Definition

Enter the package body's definition. To edit the Package Specification, click **Package** to open the Package Designer.

Hint: To customize the view of the editor and find out more features for sql editing, see Editor Advanced Features.

Result

To run the package click **Execute** on the toolbar. If the SQL statement is correct, the statement will be executed and, if the statement is supposed to return data, the **Result** tab opens with the data returned by the package. If an error occurs while executing the package, execution stops, the appropriate error message is displayed. Select the function/procedure and enter the parameter(s) if the function/procedure has input parameter(s).

Debug (Available only in Full Version)

To debug the package click **Debug** on the toolbar to launch the Oracle Debugger. Select the function/procedure and enter the parameter(s) if the function/procedure has input parameter(s).

You can add/remove breakpoints for debugging by clicking in the grey area beside each statement.

Oracle Sequences

Sequence involves creating and initializing a new special single-row table. It is usually used to generate unique identifiers for rows of a table. Click -> **Sequence** to open an object list for **Sequence**.

General Properties

Starting Value

To specify the first sequence number to be generated.

Increment

To specify which value is added to the current sequence value to create a new value. A positive value will make an ascending sequence, a negative one a descending sequence. The default value is 1.

Minimum Value

The minimum value a sequence can generate.

Maximum Value

The maximum value for the sequence.

Cache Size

To specify how many values of the sequence the database preallocates and keeps in memory for faster access. The minimum value for this parameter is 2.

No Cache

This option indicates that values of the sequence are not preallocated.

Cyclic

This option allows the sequence continues to generate values after reaching either its maximum or minimum value. After an

ascending sequence reaches its maximum value, it generates its minimum value. After a descending sequence reaches its minimum, it generates its maximum value.

Order

This option guarantees that sequence numbers are generated in order of request.

Oracle Synonyms

Synonym is an alias for any table, view, materialized view, synonym, procedure, function, package, type, Java class schema object, user-defined object type, or another synonym. Because a synonym is simply an alias, it requires no storage other than its definition in the data dictionary. Click -> **Synonym** to open an object list for **Synonym**.

General Properties

DB Link

A complete or partial database link to create a synonym for a schema object on a remote database where the object is located.

Object Schema

The schema in which the object resides.

Object Type

The object type.

Object Name

The object for which the synonym is created.

Oracle Triggers

Triggers are similar to procedures. A trigger stored in the database can include SQL and PL/SQL or Java statements to run as a unit and can invoke procedures. Click -> **Trigger** to open an object list for **Trigger**.

See Triggers for details.

You can choose the **Trigger Type**: TABLE, VIEW, SCHEMA or DATABASE.

General Properties for Table Trigger

Enabled

An enabled trigger runs its trigger action if a triggering statement is issued and the trigger restriction (if any) evaluates to true.

Table Owner

The owner of the table.

Table Name

The table you wish to create the trigger.

Compound

A compound trigger is a single trigger on a table that allows you to specify actions for each of four timing points:

Timing Point	Section
Before the triggering statement executes	BEFORE STATEMENT
After the triggering statement executes	AFTER STATEMENT
Before each row that the triggering statement affects	BEFORE EACH ROW
After each row that the triggering statement affects	AFTER EACH ROW

Note: Support from Oracle 11g or later and you can edit the SQL in Trigger Definition.

Fire

When defining a trigger, you can specify the trigger timing - whether the trigger action is to be run **Before** or **After** the triggering statement.

For Each

Oracle Database fires a **ROW** trigger once for each row that is affected by the triggering statement and fires a **STATEMENT** trigger only once when the triggering statement is issued if the optional trigger constraint is met.

When

To specify the trigger condition, which is a SQL condition that must be satisfied for the database to fire the trigger.

Insert

The trigger is activated whenever adding a row to a table or adds an element to a nested table.

Delete

The trigger is activated whenever removing a row from the table or removes an element from a nested table.

Update

The trigger is activated whenever changing a value in one of the fields selected in **Columns**.

General Properties for View Trigger

Enabled

An enabled trigger runs its trigger action if a triggering statement is issued and the trigger restriction (if any) evaluates to true.

View Owner

The owner of the view.

View Name

The view you wish to create the trigger.

Compound

To specify the Instead Of Trigger.

Nested table

To select the nested table field.

Note: Support from Oracle 11g or later and you can edit the SQL in Trigger Definition.

Insert

The trigger is activated whenever adding a row to a table or adds an element to a nested table.

Delete

The trigger is activated whenever removing a row from the table or removes an element from a nested table.

Update

The trigger is activated whenever changing a value in a row.

General Properties for Schema Trigger

Enabled

An enabled trigger runs its trigger action if a triggering statement is issued and the trigger restriction (if any) evaluates to true.

Schema

The schema of the trigger.

Fire

When defining a trigger, you can specify the trigger timing - whether the trigger action is to be run **Before** or **After** the triggering statement.

When

To specify the trigger condition, which is a SQL condition that must be satisfied for the database to fire the trigger.

Events

Choose the DDL statements that can cause the trigger to fire.

General Properties for Database Trigger

Enabled

An enabled trigger runs its trigger action if a triggering statement is issued and the trigger restriction (if any) evaluates to true.

Fire

When defining a trigger, you can specify the trigger timing - whether the trigger action is to be run **Before** or **After** the triggering statement.

When

To specify the trigger condition, which is a SQL condition that must be satisfied for the database to fire the trigger.

Events

Choose the DDL statements that can cause the trigger to fire.

Advanced Properties for Table and View Trigger

Old

Correlation names of the old nested table.

New

Correlation names of the new nested table.

Parent

Correlation names of the parent table.

Follows

To indicate that the trigger should fire after the specified triggers. Use the **Schema** dropdown list to select the schema name and **Trigger Name** dropdown list to select the trigger.

Note: Support from Oracle 11g or later.

Definition

You can edit valid SQL or procedure statements in the trigger definition inside *BEGIN* and *END*.

The **Code Outline** window displays information about the trigger. To show the **Code Outline** window, simply choose **Edit** -> **Show Code Outline** from main menu.

Note: Available only in Full Version.

Button	Description
	Refresh the code outline.
	Turn mouse over highlight on or off.
	Show the code outline in a detail way.
	Sort by type and name.
	Expand the selected item.
	Collapse the selected item.

Hint: To customize the view of the editor and find out more features for sql editing, see Editor Advanced Features.

Oracle Types

Type is an user-defined datatype that model the structure and behavior of the data in an application. An object type consists of two parts: a specification and a body. The type body always depends on its type specification. A collection type is a named varying array (varray) or a nested table type. Click -> **Type** to open an object list for **Type**.

To create an object type, simply click **Add** from the object list toolbar. Or, you can click-and-hold on **Add** to choose the type **Object** / **Collection**.

The **Code Outline** window displays information about the object type/object type body including declaration, etc. To show the **Code Outline** window, simply choose **Edit** -> **Show Code Outline** from main menu.

Note: Available only in Full Version.

Button	Description
	Refresh the code outline.
	Turn mouse over highlight on or off.
	Show the code outline in a detail way.
	Sort by type and name.
	Expand the selected item.
	Collapse the selected item.

Object Type's Definition

Enter the object type's definition. After saving the object type, you can edit the Object Type Body. Just click **New Type Body** or **Edit Type Body** to open the Type Body Designer.

Hint: To customize the view of the editor and find out more features for sql editing, see Editor Advanced Features.

Object Type Body's Definition

Enter the object type body's definition. To edit the Object Type Specification, click **Type** to open the Object Type Designer.

Hint: To customize the view of the editor and find out more features for sql editing, see Editor Advanced Features.

Collection Type

Nested table

Create a nested table type.

Varying array

Create a varray type and determine the array size of the varray type.

Data Type

Select the Oracle Database built-in datatype or user-defined type of the attribute.

Data Type Parameter(s)

Determine the corresponding data type parameters.

Oracle XML Schemas

XML Schema is a schema definition language written in XML. It can be used to describe the structure and various other semantics of conforming instance documents. Click -> **XML Schema** to open an object list for **XML Schema**.

Schema Doc

Enter a valid XML schema document under the **Schema Doc** tab.

Advanced Properties

Local

Check this to register as local schemas.

Force On Schema Registration

Check this to ignore errors generated during schema evolution.

Object Types

Check this to enable the schema compiler to generate object types.

Java Beans

Check this to enable the schema compiler to generate Java beans.

Default Tables

Check this to enable the schema compiler to generate default tables.

REGISTER_NODOCID

Check this to prevent the creation of this column if the user wishes to optimize on storage.

REGISTER_BINARYXML

Check this to register the schema for Binary XML.

REGISTER_NT_AS_IOT

Check this to store nested tables created during schema registration as index organized tables.

REGISTER_AUTO_OOL

Check this to automatically move large types out of line.

Enable Hierarchy

ENABLE_HIERARCHY_NONE	Enable hierarchy will not be called on any tables created while registering that schema.
ENABLE_HIERARCHY_CONTENTS	Enable hierarchy will be called for all tables created during schema registration with hierarchy_type as DBMS_XDBZ.ENABLE_CONTENTS.
ENABLE_HIERARCHY_RESMETADATA	Enable hierarchy will be called on all tables created during schema registration with hierarchy_type as DBMS_XDBZ.ENABLE_RESMETADATA.

Oracle Recycle Bin

Recycle bin is actually a data dictionary table containing information about dropped objects. Dropped tables and any associated objects such as indexes, constraints, nested tables, and the likes are not removed and still occupy space. They continue to count

against user space quotas, until specifically purged from the recycle bin or the unlikely situation where they must be purged by the database because of tablespace space constraints. Click -> **Recycle Bin** to open an object list for **Recycle Bin**.

To restore a table

1. Choose a table in recycle bin.
2. Click **Flashback** from the object list toolbar.

To remove an object

1. Select an object for purging in the object list.
2. Click **Purge** from the object list toolbar.
3. Confirm deleting in the dialog window.

To remove all objects

1. Control-click and select **Purge Recycle Bin** from the popup menu.
2. Confirm deleting in the dialog window.

To remove all objects of any users

1. Log in a user has the **SYSDBA** privilege.
2. Control-click and select the **Purge DBA Recycle Bin** from the popup menu.
3. Confirm deleting in the dialog window.

Oracle Directories

A directory object specifies an alias for a directory on the server file system where external binary file LOBs (BFILEs) and external table data are located. All directories are created in a single namespace and are not owned by an individual schema. Click -> **Directory** to open an object list for **Directory**.

General Properties

Path Name

Specify the full path name of the operating system directory of the server where the files are located. The path name is case sensitive.

Oracle Tablespaces

Tablespaces are the allocation of space in the database that can contain schema objects. Click -> **Tablespace** to open an object list for **Tablespace**.

General Properties

Tablespace Type

PERMANENT	A permanent tablespace contains persistent schema objects. Objects in permanent tablespaces are stored in datafiles.
TEMPORARY	A temporary tablespace contains schema objects only for the duration of a session. Objects in temporary tablespaces are stored in tempfiles.
UNDO	An undo tablespace is a type of permanent tablespace used by Oracle Database to manage undo data if you are running your database in automatic undo management mode.

Name

Set the name of the datafile/tempfile.

Size

Set the size of the datafile/tempfile.

Unit

Define the size unit of the datafile/tempfile. Specify the maximum disk space allowed for automatic extension of the datafile. Use the dropdown list K, M, G or T to specify the size in kilobytes, megabytes, gigabytes or terabytes.

Reuse

To allow Oracle to reuse an existing file.

Path

Specify the path of the datafile/tempfile.

Autoextend

To **ON** (enable) or **OFF** (disable) the automatic extension of a new or existing datafile or tempfile.

Next Size

Specify the size in bytes of the next increment of disk space to be allocated automatically when more extents are required. The default is the size of one data block. Use the dropdown list K, M, G or T to specify the size in kilobytes, megabytes, gigabytes or terabytes.

Unlimited max

Unlimited disk space that Oracle can allocate to the datafile or tempfile.

Max Size

Specify the maximum disk space allowed for automatic extension of the datafile. Use the dropdown list K, M, G or T to specify the size in kilobytes, megabytes, gigabytes or terabytes.

Storage

File Type

BIGFILE	A bigfile tablespace contains only one datafile or tempfile, which can contain up to approximately 4 billion (2^{32}) blocks. The maximum size of the single datafile or tempfile is 128 terabytes (TB) for a tablespace with 32K blocks and 32TB for a tablespace with 8K blocks.

SMALLFILE	A smallfile tablespace is a traditional Oracle tablespace, which can contain 1022 datafiles or tempfiles, each of which can contain up to approximately 4 million (2^{22}) blocks.

Minimum Extent Size

The minimum size of an extent in the tablespace. Use the dropdown list K, M, G or T to specify the size in kilobytes, megabytes, gigabytes or terabytes.

Block size

The block size for the tablespace.

Use storage option

Click **Storage Option** to set the storage options of the tablespace.

Table Compression

Use the dropdown list to select the type of compressing data segments to reduce disk use.

Manual segment management

To manage the free space of segments in the tablespace using free lists.

Extent Management

DICTIONARY	Extent management by the data dictionary.
LOCAL	Extent management by the bitmaps.

Allocation

AUTOALLOCATE	The tablespace is system managed.
UNIFORM	The tablespace is managed with uniform extents of size.

Uniform Size

The size of uniform extent. The default size is 1 megabyte. Use the dropdown list K, M, G or T to specify the size in kilobytes, megabytes, gigabytes or terabytes. If you do not specify any of the abbreviations, then the size is interpreted as bytes.

Advanced Properties

Logging

LOGGING	Log all objects within the tablspace in the redo log file.
NOLOGGING	No operations are logged.

Force logging

Oracle Database will log all changes to all objects in the tablespace except changes to temporary segments, overriding any NOLOGGING setting for individual objects.

Offline

The tablespace is unavailable immediately (offline) after creation.

Retention guarantee

Oracle Database should preserve unexpired undo data in all undo segments of tablespace even if doing so forces the failure of ongoing operations that need undo space in those segments.

Tablespace Group

To determine whether tablespace is a member of a tablespace group.

Flashback

ON	Oracle Database will save Flashback log data for this tablespace and the tablespace can participate in a FLASHBACK DATABASE operation.
OFF	Oracle Database will not save any Flashback log data for this tablespace.

Use encryption

Enable the encryption properties of the tablespace.

Algorithm

To select the encryption algorithm.

Oracle Public Database Links

Public database Link is a database link created by a *DBA* on a local database that is accessible to all users on that database. Click -> **Public Database Link** to open an object list for **Public Database Link**.

See Database Link for details.

Oracle Public Synonyms

Public synonym is a synonym owned by the special user group named *PUBLIC* and every user in a database can access it. Click -> **Public Synonym** to open an object list for **Public Synonym**.

See Synonyms for details.

PostgreSQL Objects

To start working with the server objects, you should create and open a connection. If the server is empty, you need to right-click the connection in the connection tree and choose **New Database** to create a new database.

General Properties

To create a database, you must have the **Can create database** (usecreatedb) right. Refer to Role Designer or User Designer on how to set user properties.

Name

Define the name of the database.

Owner

Define the owner for the database. If omitted, defaults to the user executing the command. Only superusers may create database owned by users other than themselves.

Encoding

Define the encoding for the database. If omitted, the default is the encoding of the template database.

Collation Order

Define the collation order (LC_COLLATE) to use in the new database. This affects the sort order applied to strings, e.g. in queries with ORDER BY, as well as the order used in indexes on text columns. The default is to use the collation order of the template database. See below for additional restrictions.

Note: Support from PostgreSQL 8.4 or later.

Character Classification

Define the character classification (LC_CTYPE) to use in the new database. This affects the categorization of characters, e.g. lower, upper and digit. The default is to use the character classification of the template database. See below for additional restrictions.

Note: Support from PostgreSQL 8.4 or later.

Template

Create the database from a template database.

Note: It is essential that the source database be idle (no data-altering transactions in progress) for the duration of the copying operation. CREATE DATABASE will check that no session (other than itself) is connected to the source database at the start of the operation, but this does not guarantee that changes cannot be made while the copy proceeds, which would result in an inconsistent copied database. Therefore, it is recommended that databases used as templates be treated as read-only.

Tablespace

Define the tablespace for the database. If omitted, defaults to pg_default.

Connection Limit

Define how many concurrent connections can be made to this database. -1 (the default) means no limit.

Note: Support from PostgreSQL 8.1 or later.

PostgreSQL Schemas

A schema is essentially a namespace: it contains named objects (tables, data types, functions, and operators) whose names may duplicate those of other objects existing in other schemas. The schema name must be distinct from any existing schema name in the current database.

General Properties

Name

The name of a schema to be created. The name cannot begin with pg_, as such names are reserved for system schemas.

Owner

The name of the user who will own the schema. If omitted, defaults to the user executing the command.

PostgreSQL Tables

Relational databases use tables to store data. All operations on data are done on the tables themselves or produce another table as the result. A table is a set of rows and columns, and their intersections are fields. From a general perspective, columns within a table describe the name and type of data that will be found by row for that column's fields. Rows within a table represent records composed of fields that are described from left to right by their corresponding column's name and type. Each field in a row is implicitly correlated with each other field in that row. Click to open an object list for **Table**.

When open a table with graphical fields, control-click a table and select **Open Table (Quick)** from the popup menu. Faster performance for opening the graphical table, as BLOB fields (images) will not be loaded until you click on the cell. If you do wish Navicat loads all your images while opening the table, click **Open** button from the object list toolbar.

You can create a table shortcut by drag the table out. It provides a convenient way for you to open your table for entering data directly without activating the main Navicat.

To empty a table, control-click the selected table and choose **Empty Table** from the popup menu. This option is only applied when you wish to clear all the existing records without resetting the auto-increment value. To reset the auto-increment value while emptying your table, use **Truncate Table** below.

PostgreSQL Table Fields

In the **Fields** tab, just simply click a field for editing. By using the field toolbar, you can create new and drop the selected field. To search a field name, choose **Edit** -> **Find** -> **Find** or press CMD-F.

Use the **Name** edit box to set the field name. Note that the name of the field must be unique among all the field names in the table.

The **Type** dropdown list defines the type of the field data. See PostgreSQL Data Types for details.

Use the **Length** edit box to define the length of the field and use **Decimals** edit box to define the number of digits after the decimal point (the scale) for Floating Point data type.

Note: Be careful when shortening the field length as losing data might be caused.

Dimensions

Set the dimensions of array specifiers.

Not Null

Check this box to not allow the NULL values for the field.

Key

A Primary Key is a single field or combination of fields that uniquely defines a record. None of the fields that are part of the primary key can contain a null value.

Field's Properties

Note: The following options depend on the field type you are chosen.

Default Value

Set the default value for the field.

Comment

Set any optional text describing the current field.

Collation

Set the collation of the column (which must be of a collatable data type). If not specified, the column data type's default collation is used.

Note: Support from PostgreSQL 9.1 or later.

Schema

Set the schema for the field type.

User-defined Type

Set the type for the field.

PostgreSQL Table Indexes

Indexes are primarily used to enhance database performance (though inappropriate use can result in slower performance). An index field can be an expression computed from the values of one or more columns of the table row. This feature can be used to obtain fast access to data based on some transformation of the basic data.

In the **Indexes** tab, just simply click an index field for editing. By using the index toolbar, you can create new, edit and delete the selected index field.

Use the **Name** edit box to set the index name. No schema name can be included here; the index is always created in the same schema as its parent table.

To include field(s) in the index, just simply click **Fields** to open the editor for editing.

Note: Some of field types do not allow indexing by several fields.

The **Index Type** dropdown list defines the type of the table index.

Unique

Makes index unique, causes the system to check for duplicate values in the table when the index is created (if data already exist) and each time data is added.

Cluster

CLUSTER instructs PostgreSQL to cluster the table specified by tablename based on the index specified by indexname. The index must already have been defined on tablename.

When a table is clustered, PostgreSQL remembers on which index it was clustered. The form CLUSTER tablename reclusters the table on the same index that it was clustered before.

Tablespace

The tablespace in which to create the index.

Constraints

If you wish to create partial index, enter constraint condition in this edit box. A partial index is an index that contains entries for only a portion of a table, usually a portion that is more useful for indexing than the rest of the table.

Fill Factor

The fillfactor for an index.

Note: Support from PostgreSQL 8.2 or later.

Comment

Define the comment for the index.

Fields Editor

Select the field(s) from the **Field** list.

Collation

Choose the collation for the index.

Note: Support from PostgreSQL 9.1 or later.

Order

Specify the sort order: ASC or DESC.

Nulls

Specify that nulls sort before(FIRST)/after(LAST) non-nulls.

PostgreSQL Table Foreign Keys

A foreign key specifies that the values in a column (or a group of columns) must match the values appearing in some row of another table. We say this maintains the referential integrity between two related tables.

In the **Foreign Keys** tab, just simply click a foreign key field for editing. By using the foreign key toolbar, you can create new, edit and delete the selected foreign key field.

Use the **Name** edit box to enter a name for the new key.

Use the **Referenced Schema** and **Referenced Table** dropdown lists to select a foreign schema and table respectively.

To include field(s)/referenced field(s) to the key, click **Fields** or **Referenced Fields** to open the editor(s) for editing.

The **On Delete** and **On Update** dropdown list define the type of the actions to be taken.

CASCADE	Delete any rows referencing the deleted row, or update the value of the referencing column to the new value of the referenced column, respectively.
SET NULL	Set the referencing column(s) to null.
NO ACTION	Produce an error indicating that the deletion or update would create a foreign key constraint violation. If the constraint is deferred, this error will be produced at constraint check time if there still exist any referencing rows. This is the default action.
RESTRICT	Produce an error indicating that the deletion or update would create a foreign key constraint violation. This is the same as NO ACTION except that the check is not deferrable.

Related topic:

Foreign Keys Data Selection

PostgreSQL Table Uniques

Unique constraints ensure that the data contained in a column or a group of columns is unique with respect to all the rows in the table.

In the **Uniques** tab, just simply click an unique field for editing. By using the unique toolbar, you can create new, edit and delete the selected unique field.

Use the **Name** edit box to set the unique name.

To set field(s) as unique, click **Fields** to open the editor(s) for editing.

Index Tablespace

Allow setting a tablespace different from the default tablespace.

Fill Factor

The fillfactor for an unique is a percentage between 10 and 100. 100 (complete packing) is the default.

Note: Support from PostgreSQL 8.2 or later.

Comment

Define the comment for the unique.

PostgreSQL Table Checks

A check constraint is the most generic constraint type. It allows you to specify that the value in a certain column must satisfy a Boolean (truth-value) expression.

In the **Checks** tab, just simply click a check field for editing. By using the check toolbar, you can create new, edit and delete the selected check field.

Use the **Name** edit box to set the check name.

Expression

Set the condition for checking, e.g. "field_name1 > 0 AND field_name2 > field_name1" in the **Expression** edit box. A check constraint specified as a column constraint should reference that column's value only, while an expression appearing in a table constraint may reference multiple columns.

Comment

Enter the comment for the check.

PostgreSQL Table Excludes

A exclude constraint guarantees that if any two rows are compared on the specified column(s) or expression(s) using the specified operator(s), not all of these comparisons will return TRUE.

In the **Excludes** tab, just simply click an exclude field for editing. By using the exclude toolbar, you can create new, edit and delete the selected exclude field.

Note: Exclude is supported from PostgreSQL 9.0 or later.

Use the **Name** edit box to set the exclude name.

Index Method

The name of the index access method to be used.

Exclude Elements

Choose the element(s) to be excluded and specify the operator(s).

Index Tablespace

The tablespace in which to create the index.

Fill Factor

The fillfactor for an index is a percentage that determines how full the index method will try to pack index pages.

Predicate

Specify an exclusion constraint on a subset of the table.

Comment

Define the comment for the exclude.

PostgreSQL Table Rules

The PostgreSQL rule system allows one to define an alternate action to be performed on insertions, updates, or deletions in database tables. Roughly speaking, a rule causes additional commands to be executed when a given command on a given table is executed.

Note: You must be the owner of a table to create or change rules for it.

In the **Rules** tab, just simply click a rule field for editing. By using the rule toolbar, you can create new, edit and delete the selected rule field.

Use the **Name** edit box to set the rule name. This must be distinct from the name of any other rule for the same table. Multiple rules on the same table and same event type are applied in alphabetical name order.

Event Type

The event is one of *SELECT*, *INSERT*, *UPDATE*, or *DELETE*.

Do

also	This indicates that the commands should be executed in addition to the original command.
instead	This indicates that the commands should be executed instead of the original command.

Condition

Any SQL conditional expression (returning boolean). The condition expression may not refer to any tables except NEW and OLD, and may not contain aggregate functions.

Commands

The command or commands that make up the rule action. Valid commands are *SELECT*, *INSERT*, *UPDATE*, *DELETE*, or *NOTIFY*.

Within condition and command, the special table names NEW and OLD may be used to refer to values in the referenced table. NEW is valid in ON INSERT and ON UPDATE rules to refer to the new row being inserted or updated. OLD is valid in ON UPDATE and ON DELETE rules to refer to the existing row being updated or deleted.

Comment

Define the comment for the rule.

PostgreSQL Table Triggers

A trigger is a specification that the database should automatically execute a particular function whenever a certain type of operation is performed. Triggers can be defined to execute either before or after any INSERT, UPDATE, or DELETE operation, either once per modified row, or once per SQL statement.

In the **Triggers** tab, just simply click a trigger field for editing. By using the trigger toolbar, you can create new, edit and delete the selected trigger field.

Note: To create a trigger on a table, the user must have the TRIGGER privilege on the table.

Use the **Name** edit box to set the trigger name. This must be distinct from the name of any other trigger for the same table.

For Each

This specifies whether the trigger procedure should be fired once for every row affected by the trigger event, or just once per SQL statement.

Fires

Define the trigger action time. It can be **before** or **after** to indicate that the trigger activates before or after the statement that activated it.

Insert

The trigger is activated whenever a new row is inserted into the table.

Update

The trigger is activated whenever a row is modified.

Delete

The trigger is activated whenever a row is deleted from the table.

Update Of

Specify a list of columns. The trigger will only fire if at least one of the listed columns is mentioned as a target of the UPDATE command.

Note: Support from PostgreSQL 9.1 or later.

Condition

Specify a Boolean WHEN condition, which will be tested to see whether the trigger should be fired.

Note: Support from PostgreSQL 9.0 or later.

Function Schema and **Function**

A user-supplied function that is declared as taking no arguments and returning type trigger, which is executed when the trigger fires.

Arguments

An optional comma-separated list of arguments to be provided to the function when the trigger is executed. The arguments are literal string constants. Simple names and numeric constants may be written here, too, but they will all be converted to strings. Please check the description of the implementation language of the trigger function about how the trigger arguments are accessible within the function; it may be different from normal function arguments.

Comment

Define the comment for the trigger.

PostgreSQL Table Options

Owner

Define the user to own this table.

Tablespace

Define a tablespace different from the default tablespace to create a table.

Note: Support from PostgreSQL 8.0 or later.

Has oids

Check this option if you want to specify whether rows of the new table should have OIDs (object identifiers) assigned to them.

Inherits from

This option specifies a list of tables from which the new table automatically inherits all columns. Use of inheritance creates a persistent relationship between the new child table and its parent table(s). Schema modifications to the parent(s) normally propagate to children as well, and by default the data of the child table is included in scans of the parent(s).

Select the available table(s) from the list.

Fill Factor

The fillfactor for a table is a percentage between 10 and 100. 100 (complete packing) is the default. When a smaller fillfactor is specified, INSERT operations pack table pages only to the indicated percentage; the remaining space on each page is reserved for updating rows on that page. This gives UPDATE a chance to place the updated copy of a row on the same page as the original, which is more efficient than placing it on a different page. For a table whose entries are never updated, complete packing is the best choice, but in heavily updated tables smaller fillfactors are appropriate.

Note: Support from PostgreSQL 8.2 or later.

PostgreSQL Views

Views are useful for allowing users to access a set of relations (tables) as if it were a single table, and limiting their access to just that. Views can also be used to restrict access to rows (a subset of a particular table). Click to open an object list for **View**.

You can create a view shortcut by drag the view out. It provides a convenient way for you to open your view without activating the main Navicat.

Button	Description
Preview	Preview and/or Explain the view.

Note: You can choose to show the Result tab below the editor or in a new tab by selecting **Edit** -> **Show Result** -> **Below Query Editor** or **In a New Tab**.

View Builder (Available only in Full Version)

View Builder allows you to build views visually. It allows you to create and edit views without knowledge of SQL. See Query Builder for details.

View Editor

You can edit the view definition as SQL statement (SELECT statement it implements).

Hint: To customize the view of the editor and find out more features for sql editing, see Editor Advanced Features.

Rules

Use the **Name** edit box to set the rule name.

Event Type

The event is one of *SELECT*, *INSERT*, *UPDATE*, or *DELETE*.

Do

also	This indicates that the commands should be executed in addition to the original command.
instead	This indicates that the commands should be executed instead of the original command.

Condition

Any SQL conditional expression (returning boolean). The condition expression may not refer to any tables except NEW and OLD, and may not contain aggregate functions.

Commands

The command or commands that make up the rule action. Valid commands are *SELECT*, *INSERT*, *UPDATE*, *DELETE*, or *NOTIFY*.

Within condition and command, the special table names NEW and OLD may be used to refer to values in the referenced table. NEW is valid in ON INSERT and ON UPDATE rules to refer to the new row being inserted or updated. OLD is valid in ON UPDATE and ON DELETE rules to refer to the existing row being updated or deleted.

Comment

Define the comment for the rule.

View Viewer

View Viewer displays the view data as a grid. Data can be displayed in two modes: **Grid View** and **Form View**. See Table Viewer for details.

PostgreSQL Functions

PostgreSQL provides four kinds of functions:

- query language functions (functions written in SQL)
- procedural language functions (functions written in, for example, PL/Tcl or PL/pgSQL)
- internal functions
- C-language functions

Every kind of function can take base types, composite types, or combinations of these as arguments (parameters). In addition, every kind of function can return a base type or a composite type. Many kinds of functions can take or return certain pseudo-types (such as polymorphic types), but the available facilities vary. Click to open an object list for **Function**.

Properties

Owner

The owner of the function.

Language

The name of the language that the function is implemented in. May be SQL, C, internal, or the name of a user-defined procedural language. For backward compatibility, the name may be enclosed by single quotes.

Return Type Schema and **Return Type**

The return type of the function.

Volatility

These attributes inform the query optimizer about the behavior of the function. At most one choice may be specified. If none of these appear, VOLATILE is the default assumption.

IMMUTABLE	The function cannot modify the database and always returns the same result when given the same argument values; that is, it does not do database lookups or otherwise use information not directly present in its argument list. If this option is given, any call of the function with all-constant arguments can be immediately replaced with

	the function value.
STABLE	The function cannot modify the database, and that within a single table scan it will consistently return the same result for the same argument values, but that its result could change across SQL statements. This is the appropriate selection for functions whose results depend on database lookups, parameter variables (such as the current time zone), etc. Also note that the current_timestamp family of functions qualify as stable, since their values do not change within a transaction.
VOLATILE	The function value can change even within a single table scan, so no optimizations can be made. Relatively few database functions are volatile in this sense; some examples are random(), currval(), timeofday(). But note that any function that has side-effects must be classified volatile, even if its result is quite predictable, to prevent calls from being optimized away; an example is setval().

Security

INVOKER	Indicate that the function is to be executed with the privileges of the user that calls it.
DEFINER	Specify that the function is to be executed with the privileges of the user that created it.

Returns set

Indicate that the function will return a set of items, rather than a single item.

Strict

Indicate that the function always returns null whenever any of its arguments are null. If this parameter is specified, the function is not executed when there are null arguments; instead a null result is assumed automatically.

Parameter

Define function parameter.

Estimated Cost

A positive number giving the estimated execution cost for the function, in units of cpu_operator_cost. If the function returns a set, this is the cost per returned row. If the cost is not specified, 1 unit is assumed for C-language and internal functions, and 100 units for functions in all other languages. Larger values cause the planner to try to avoid evaluating the function more often than necessary.

Note: Support from PostgreSQL 8.3 or later.

Estimated Rows

A positive number giving the estimated number of rows that the planner should expect the function to return. This is only allowed when the function is declared to return a set. The default assumption is 1000 rows.

Note: Support from PostgreSQL 8.3 or later.

Configuration Parameter

The specified configuration parameter to be set to the specified value when the function is entered, and then restored to its prior value when the function exits.

Note: Support from PostgreSQL 8.3 or later.

Definition

Definition consists of a valid SQL procedure statement. This can be a simple statement such as *SELECT* or *INSERT*, or it can be a compound statement written using *BEGIN* and *END*. Compound statements can contain declarations, loops, and other control structure statements.

Hint: To customize the view of the editor and find out more features for sql editing, see Editor Advanced Features.

Result

To run the function, click **Execute** on the toolbar. If the SQL statement is correct, the statement will be executed and, if the statement is supposed to return data, the **Result** tab opens with the data returned by the function. If an error occurs while executing the function, execution stops, the appropriate error message is displayed. If the function requires input parameter, the **Input Parameters** box will popup.

PostgreSQL Aggregates

Aggregate functions in PostgreSQL are expressed as state values and state transition functions. That is, an aggregate can be defined in terms of state that is modified whenever an input item is processed. To define a new aggregate function, one selects a data type for the state value, an initial value for the state, and a state transition function. The state transition function is just an ordinary function that could also be used outside the context of the aggregate. A final function can also be specified, in case the desired result of the aggregate is different from the data that needs to be kept in the running state value. Click -> **Aggregate** to open an object list for **Aggregate**.

General Properties

Owner

The owner of the aggregate function.

Input Type

An input data type on which this aggregate function operates.

Note: Support from PostgreSQL 8.2 or later. For versions below 8.2, just select the **Schema** and **Type** from the dropdown lists.

State Type Schema and **State type**

The data type for the aggregate's state value.

State Function Schema and **State function**

The state transition function to be called for each input row. For an N-argument aggregate function, the state function must take N+1 arguments, the first being of type *state_data_type* and the rest matching the declared input data type(s) of the aggregate. The function must return a value of type *state_data_type*. This function takes the current state value and the current input data value(s), and returns the next state value.

Final Function Schema and **Final function**

The final function called to compute the aggregate's result after all input rows have been traversed. The function must take a single argument of type *state_data_type*. The return data type of the aggregate is defined as the return type of this function. If final function is not specified, then the ending state value is used as the aggregate's result, and the return type is *state_data_type*.

Initial Condition

The initial setting for the state value. This must be a string constant in the form accepted for the data type *state_data_type*. If not specified, the state value starts out null.

Sort Operator Schema and **Sort Operator**

The associated sort operator for a MIN- or MAX-like aggregate. The operator is assumed to have the same input data types as the aggregate (which must be a single-argument aggregate).

Note: Support from PostgreSQL 8.1 or later.

PostgreSQL Conversions

Conversion defines a new conversion between character set encodings. Conversion names may be used in the convert function to specify a particular encoding conversion. Also, conversions that are marked DEFAULT can be used for automatic encoding conversion between client and server. For this purpose, two conversions, from encoding A to B and from encoding B to A, must be defined. Click -> **Conversion** to open an object list for **Conversion**.

General Properties

Owner

The owner of the conversion function.

Source Encoding

The source encoding name.

Target Encoding

The destination encoding name.

Schema of Function and **Function**

The function used to perform the conversion. The function name may be schema-qualified. If it is not, the function will be looked up in the path.

The function must have the following signature:

conv_proc(

integer, -- source encoding ID

integer, -- destination encoding ID

cstring, -- source string (null terminated C string)

internal, -- destination (fill with a null terminated C string)

integer -- source string length

) RETURNS void;

Default

Check this box to indicate that this conversion is the default for this particular source to destination encoding. There should be only one default encoding in a schema for the encoding pair.

PostgreSQL Domains

A domain is essentially a data type with optional constraints (restrictions on the allowed set of values). The user who defines a domain becomes its owner. Domains are useful for abstracting common constraints on fields into a single location for maintenance. For example, several tables might contain email address columns, all requiring the same *CHECK* constraint to verify the address syntax. Define a domain rather than setting up each table's constraint individually. Click -> **Domain** to open an object list for **Domain**.

General Properties

Underlying Type Category

Choose the underlying data type category: **Base Type**, **Composite Type**, **Enum Type** and **Domain**.

Note: Support from PostgreSQL 8.2 or later.

Underlying Type Schema

Select schema of the underlying data type.

Underlying Type

Select the underlying data type of the domain from the drop-down list.

Dimensions

The dimensions of array specifiers.

Length and **Scale**

Use the **Length** edit box to define the length of the field and use **Scale** edit box to define the number of digits after the decimal point. (if required for the selected data type)

Default

The *DEFAULT* clause specifies a default value for columns of the domain data type. The value is any variable-free expression (but subqueries are not allowed). The data type of the default expression must match the data type of the domain. If no default value is specified, then the default value is the null value.

The default expression will be used in any insert operation that does not specify a value for the column. If a default value is defined for a particular column, it overrides any default associated with the domain. In turn, the domain default overrides any default value associated with the underlying data type.

Not Null

Values of this domain are not allowed to be null.

Owner

The owner of the domain function. The user who defines a domain becomes its owner.

Check

The **Check** tab is provided for managing domain checks. It allows you to create new, edit, or delete the selected check.

CHECK clauses specify integrity constraints or tests which values of the domain must satisfy. Each constraint must be an expression producing a Boolean result. It should use the key word *VALUE* to refer to the value being tested.

PostgreSQL Indexes

Index provides a faster access path to table data. It is created using one or more columns of a table to speed SQL statement execution on that table. Click -> **Index** to open an object list for **Index**.

General Properties

Type

Define the type of the index.

Unique

Makes index unique, causes the system to check for duplicate values in the table when the index is created (if data already exist) and each time data is added.

Table Name

The name (possibly schema-qualified) of the table to be indexed.

Field

The name of a column of the table.

Collation

Choose the collation for the index.

Note: Support from PostgreSQL 9.1 or later.

Order *(only for btree index)*

Specify the sort order: ASC or DESC.

Nulls *(only for btree index)*

Specify that nulls sort before(FIRST)/after(LAST) non-nulls.

Constraint

If you wish to create partial index, enter constraint condition in this tab. A partial index is an index that contains entries for only a portion of a table, usually a portion that is more useful for indexing than the rest of the table.

Advanced Properties

Cluster

CLUSTER instructs PostgreSQL to cluster the table specified by tablename based on the index specified by indexname. The index must already have been defined on tablename.

When a table is clustered, PostgreSQL remembers on which index it was clustered. The form CLUSTER tablename reclusters the table on the same index that it was clustered before.

Concurrently

When this option is used, PostgreSQL will build the index without taking any locks that prevent concurrent inserts, updates, or deletes on the table; whereas a standard index build locks out writes (but not reads) on the table until it's done.

Tablespace

The tablespace in which to create the index.

Fill Factor (%)

The fillfactor for an index is a percentage that determines how full the index method will try to pack index pages.

Fast Update *(only for gin index)*

This setting controls usage of the fast update technique.

Note: Support from PostgreSQL 8.4 or later.

Buffering *(only for gist index)*

This setting controls usage of the fast update technique.

PostgreSQL Operators

PostgreSQL supports left unary, right unary, and binary operators. Operators can be overloaded. At least one of *LEFTARG* and *RIGHTARG* must be defined. For binary operators, both must be defined. For right unary operators, only *LEFTARG* should be defined, while for left unary operators only *RIGHTARG* should be defined. Click -> **Operator** to open an object list for **Operator**.

Note: *LEFTARG* = Left type; *RIGHTARG* = Right type.

General Properties

Owner

The owner of the operator function.

Schema of Left Type and **Left Type**

The data type of the operator's left operand, if any. This option would be omitted for a left-unary operator.

Schema of Right Type and **Right Type**

The data type of the operator's right operand, if any. This option would be omitted for a right-unary operator.

Schema of Operator Function and **Operator Function**

The function used to implement this operator.

Advanced Properties

Schema of Restrict Function and **Restrict Function**

The restriction selectivity estimator function for this operator.

Schema of Join Function and **Join Function**

The join selectivity estimator function for this operator.

Schema of Commutator and **Commutator**

The commutator of this operator.

Schema of Negator and **Negator**

The negator of this operator.

Hash join

The operator can support a hash join if this option on.

Merge join

The operator can support a merge join if this option on.

Additional Advanced Properties for PostgreSQL Version below 8.3

Schema of Left Sort Operator and **Left Sort Operator**

If this operator can support a merge join, the left sort operator that sorts the left-hand data type of this operator.

Schema of Right Sort Operator and **Right Sort Operator**

If this operator can support a merge join, the right sort operator that sorts the right-hand data type of this operator.

Schema of Less Than Operator and **Less Than Operator**

If this operator can support a merge join, the less-than operator that compares the input data types of this operator.

Schema of Greater Than Operator and **Greater Than Operator**

If this operator can support a merge join, the greater-than operator that compares the input data types of this operator.

PostgreSQL Operator Classes

An operator class defines how a particular data type can be used with an index. The operator class specifies that certain operators will fill particular roles or "strategies" for this data type and this index method. The operator class also specifies the support procedures to be used by the index method when the operator class is selected for an index column. All the operators and functions

used by an operator class must be defined before the operator class is created. Click -> **Operator Class** to open an object list for **Operator Class**.

Note: Two operator classes in the same schema can have the same name only if they are for different index methods.

General Properties

Owner

The owner of the operator class function.

Schema of Data Type and **Data Type**

The column data type that this operator class is for.

Index Method

The name of the index method this operator class is for.

Schema of Storage Type and **Storage type**

The data type actually stored in the index. Normally this is the same as the column data type, but some index methods (*GIN* and *GiST* for now) allow it to be different. The *STORAGE* clause must be omitted unless the index method allows a different type to be used.

Operator Family

The name of the existing operator family to add this operator class to. If not specified, a family named the same as the operator class is used (creating it, if it doesn't already exist).

Note: Support from PostgreSQL 8.3 or later.

Default operator

With this option selected, the operator class will become the default operator class for its data type. At most one operator class can be the default for a specific data type and index method.

Operators

Strategy No.

The index method's strategy number for an operator associated with the operator class.

Schema of operator and **Operator Prototype**

The operator associated with the operator class.

Recheck

With this option selected, the index is "lossy" for this operator, and so the rows retrieved using the index must be rechecked to verify that they actually satisfy the qualification clause involving this operator.

Note: Before PostgreSQL 8.4, the OPERATOR clause could include a RECHECK option. This is no longer supported because whether an index operator is "lossy" is now determined on-the-fly at runtime. This allows efficient handling of cases where an operator might or might not be lossy.

Functions

Support No.

The index method's support procedure number for a function associated with the operator class.

Schema of function and **Function Prototype**

The function that is an index method support procedure for the operator class.

PostgreSQL Sequences

Sequence involves creating and initializing a new special single-row table. It is usually used to generate unique identifiers for rows of a table. Click -> **Sequence** to open an object list for **Sequence**.

General Properties

Owner

The owner of the sequence function.

Starting Value

The starting value of the sequence.

Increment

Specify which value is added to the current sequence value to create a new value. A positive value will make an ascending sequence, a negative one a descending sequence. The default value is 1.

Minimum Value

Determine the minimum value a sequence can generate. If **No minimum value** is checked, then defaults will be used.

Maximum Value

Determine the maximum value for the sequence. If **No maximum value** is checked, then default values will be used.

Cache Size

Specify how many sequence numbers are to be preallocated and stored in memory for faster access.

Cyclic

This option allows the sequence to wrap around when the maxvalue or minvalue has been reached by an ascending or descending sequence respectively. If the limit is reached, the next number generated will be the minvalue maxvalue, respectively. Otherwise, any calls to nextval after the sequence has reached its maximum value will return an error.

Add owned by

Choose the **Owned By Table** and **Owned By Column** so that the sequence is associated with a specific table column, such that if that column (or its whole table) is dropped, the sequence will be automatically dropped as well. The specified table must have the same owner and be in the same schema as the sequence.

Note: Support from PostgreSQL 8.2 or later.

PostgreSQL Triggers

Triggers are database operations that are automatically performed when a specified database event occurs. Click -> **Trigger** to open an object list for **Trigger**.

See Triggers for details.

General Properties

Constraint

Check this box to create a constraint trigger.

Trigger Type

Choose the type of the trigger: Table or View.

Note: Support from PostgreSQL 9.0 or later.

Table Name or **View Name**

Choose a table or view.

Before

The trigger can be specified to fire before the operation is attempted on a row.

After

The trigger can be specified to fire after the operation is attempted on a row.

Instead Of

The trigger can be specified to fire instead of the operation is attempted on a row.

Insert

The trigger is activated whenever a new row is inserted.

Update

The trigger is activated whenever a row is modified.

Delete

The trigger is activated whenever a row is deleted.

Truncate

Trigger defined to fire for TRUNCATE.

Update Of columns

Specify a list of columns. The trigger will only fire if at least one of the listed columns is mentioned as a target of the UPDATE command.

Statement Level

Specify the trigger procedure should be fired once per SQL statement.

Row Level

Specify the trigger procedure should be fired once for every row affected by the trigger event.

Note: Support from PostgreSQL 9.0 or later.

Function Schema and **Function Name**

A user-supplied function that is declared as taking no arguments and returning type trigger, which is executed when the trigger fires.

Function Arguments

An optional comma-separated list of arguments to be provided to the function when the trigger is executed. The arguments are literal string constants. Simple names and numeric constants may be written here, too, but they will all be converted to strings. Please check the description of the implementation language of the trigger function about how the trigger arguments are accessible within the function; it may be different from normal function arguments.

Condition

Specify a Boolean condition, which will be tested to see whether the trigger should be fired.

Constraint

Deferrable

The constraint can be deferred.

Initially Immediate

The constraint is checked after each statement.

Initially Deferred

The constraint is checked only at the end of the transaction.

Referenced Table Schema and **Referenced Table Name**

The schema and the name of another table referenced by the constraint.

PostgreSQL Trigger Functions

Trigger Function can be created with PL/pgSQL and referenced within a PostgreSQL trigger definition. The term "trigger function" is a simply a way of referring to a function that is intended to be invoked by a trigger. Triggers define operations that are performed when a specific event occurs within the database. A PL/pgSQL trigger function can be referenced by a trigger as the operation to be performed when the trigger's event occurs.

The definition of a trigger and the definition of its associated trigger function are two different things. A trigger is defined with the SQL *CREATE TRIGGER* command, whereas trigger functions are defined using the SQL *CREATE FUNCTION* command. Click -> **Trigger Function** to open an object list for **Trigger Function**.

See Triggers for details.

Advanced Properties

Owner

The owner of the trigger function.

Language

The name of the language that the function is implemented in. May be C, internal, or the name of a user-defined procedural language. For backward compatibility, the name may be enclosed by single quotes.

Return Type Schema and **Return Type**

The return type of the trigger function.

Hint: To customize the view of the editor and find out more features for sql editing, see Editor Advanced Features.

Volatility

These attributes inform the query optimizer about the behavior of the function. At most one choice may be specified. If none of these appear, VOLATILE is the default assumption.

IMMUTABLE	The function cannot modify the database and always returns the same result when given the same argument values; that is, it does not do database lookups or otherwise use information not directly present in its argument list. If this option is given, any call of the function with all-constant arguments can be immediately replaced with the function value.
STABLE	The function cannot modify the database, and that within a single table scan it will consistently return the same result for the same argument values, but that its result could change across SQL statements. This is the appropriate selection for functions whose results depend on database lookups, parameter variables (such as the current time zone), etc. Also note that the current_timestamp family of functions qualify as stable, since their values do not change within a transaction.
VOLATILE	The function value can change even within a single table scan, so no optimizations can be made. Relatively few database functions are volatile in this sense; some examples are random(), currval(), timeofday(). But note that any function that has side-effects must be classified volatile, even if its result is quite predictable, to prevent calls from being optimized away; an example is setval().

Security

INVOKER	Indicate that the function is to be executed with the privileges of the user that calls it.
DEFINER	Specify that the function is to be executed with the privileges of the user that created it.

Returns set

Indicate that the function will return a set of items, rather than a single item.

Strict

Indicate that the function always returns null whenever any of its arguments are null. If this parameter is specified, the function is not executed when there are null arguments; instead a null result is assumed automatically.

Parameter

Define trigger function parameter.

Estimated Cost

A positive number giving the estimated execution cost for the function, in units of cpu_operator_cost. If the function returns a set, this is the cost per returned row. If the cost is not specified, 1 unit is assumed for C-language and internal functions, and 100 units for functions in all other languages. Larger values cause the planner to try to avoid evaluating the function more often than necessary.

Note: Support from PostgreSQL 8.2 or later.

Estimated Rows

A positive number giving the estimated number of rows that the planner should expect the function to return. This is only allowed when the function is declared to return a set.

Note: Support from PostgreSQL 8.2 or later.

Configuration Parameter

The specified configuration parameter to be set to the specified value when the function is entered, and then restored to its prior value when the function exits.

Note: Support from PostgreSQL 8.2 or later.

Definition

Definition consists of a valid SQL procedure statement. This can be a simple statement such as *SELECT* or *INSERT*, or it can be a compound statement written using *BEGIN* and *END*. Compound statements can contain declarations, loops, and other control structure statements.

PostgreSQL Types

Type registers a new data type for use in the current database. If a schema name is given then the type is created in the specified schema. Otherwise it is created in the current schema. The type name must be distinct from the name of any existing type or domain in the same schema. (Because tables have associated data types, the type name must also be distinct from the name of any existing table in the same schema.) Click -> **Type** to open an object list for **Type**.

Base types are those, like int4, that are implemented below the level of the SQL language (typically in a low-level language such as C). They generally correspond to what are often known as abstract data types. PostgreSQL can only operate on such types through functions provided by the user and only understands the behavior of such types to the extent that the user describes them. Base types are further subdivided into scalar and array types. For each scalar type, a corresponding array type is automatically created that can hold variable-size arrays of that scalar type.

Composite types, or row types, are created whenever the user creates a table; it's also possible to define a "stand-alone" composite type with no associated table. A composite type is simply a list of base types with associated field names. A value of a composite type is a row or record of field values. The user can access the component fields from SQL queries.

Enumerated (Enum) types are data types that are comprised of a static, predefined set of values with a specific order. They are equivalent to the enum types in a number of programming languages. An example of an enum type might be the days of the week, or a set of status values for a piece of data.

Note: Enum Type was added in PostgreSQL 8.3.

General Properties for Base Type

Input Schema and **Input**

The function that converts data from the type's external textual form to its internal form.

Output Schema and **Output**

The function that converts data from the type's internal form to its external textual form.

Internal Length

A numeric constant that specifies the length in bytes of the new type's internal representation. The default assumption is that it is variable-length.

Variable

Check this option if the type length is unknown.

Pass by value

Indicate that values of this data type are passed by value rather than by reference.

Alignment

The storage alignment requirement of the data type. If specified, it must be char, int2, int4, or double; the default is int4.

Storage

The storage strategy for the data type. If specified, must be plain, external, extended, or main; the default is plain.

Element Type

The type being created is an array; this specifies the type of the array elements.

Default Value

The default value for the data type. If this is omitted, the default is null.

Delimiter

The delimiter character to be used between values in arrays made of this type.

Advanced Properties for Base Type

The **Advanced** tab is supported from PostgreSQL 7.4 or later.

Owner

The owner of the type.

Note: Support from PostgreSQL 8.0 or later.

Receive Schema and **Receive**

The function that converts data from the type's external binary form to its internal form.

Send Schema and **Send**

The function that converts data from the type's internal form to its external binary form.

Analyze Schema and **Analyze**

The function that performs statistical analysis for the data type.

Note: Support from PostgreSQL 8.0 or later.

Type Modifier Input Schema and **Type Modifier Input**

The function that converts an array of modifier(s) for the type into internal form.

Note: Support from PostgreSQL 8.3 or later.

Type Modifier Output Schema and **Type Modifier Output**

The function that converts the internal form of the type's modifier(s) to external textual form.

Note: Support from PostgreSQL 8.3 or later.

General Properties for Composite Type

Name

The name of an attribute (column) for the composite type.

Type

The name of an existing data type to become a column of the composite type.

Length and **Decimals**

Use the **Length** edit box to define the length of the field and use **Decimals** edit box to define the number of digits after the decimal point. (if required for the selected data type)

Dimensions

The dimensions of array specifiers.

Advanced Properties for Composite Type

Owner

The owner of the type.

Note: Support from PostgreSQL 8.0 or later.

General Properties for Enum Type

Label

A string literal representing the textual label associated with one value of an enum type.

Advanced Properties for Enum Type

Owner

The owner of the type.

PostgreSQL Tablespaces

A tablespace allows superusers to define an alternative location on the file system where the data files containing database objects (such as tables and indexes) may reside. Click -> **Tablespace** to open an object list for **Tablespace**.

Note: Tablespace was added in PostgreSQL 8.0.

The **Comment** tab is supported from PostgreSQL 8.2 or later.

General Properties

Owner

The name of the user who will own the tablespace. If omitted, defaults to the user executing the command. Only superusers may create tablespaces, but they can assign ownership of tablespaces to non-superusers.

Location

The directory that will be used for the tablespace. The directory must be empty and must be owned by the PostgreSQL system user. The directory must be specified by an absolute path name.

PostgreSQL Casts

A cast specifies how to perform a conversion between two data types. Click -> **Cast** to open an object list for **Cast**.

Note: The **Comment** tab is supported from PostgreSQL 8.0 or later.

General Properties

Schema of Source Type and **Source Type**

The schema and name of the source data type of the cast.

Schema of Target Type and **Target Type**

The schema and name of the target data type of the cast.

Schema of Function and **Function**

The function used to perform the cast. The function name may be schema-qualified. If it is not, the function will be looked up in the schema search path. The function's result data type must match the target type of the cast.

If no function is specify, indicates that the source type and the target type are binary compatible, so no function is required to perform the cast.

Assignment

Indicate that the cast can be invoked implicitly in assignment contexts.

Implicit

Indicate that the cast may be invoked implicitly in any context.

PostgreSQL Languages

Language can register a new procedural language with a PostgreSQL database. Subsequently, functions and trigger procedures can be defined in this new language. The user must have the PostgreSQL superuser privilege to register a new language. Click -> **Language** to open an object list for **Language**.

General Properties

Owner

The owner of the language.

Schema of Handler and **Handler**

Call Handler is the name of a previously registered function that will be called to execute the procedural language functions. The call handler for a procedural language must be written in a compiled language such as C with version 1 call convention and registered with PostgreSQL as a function taking no arguments and returning the *language_handler* type, a placeholder type that is simply used to identify the function as a call handler.

Schema of Validator and **Validator**

Validator function is the name of a previously registered function that will be called when a new function in the language is created, to validate the new function. If no validator function is specified, then a new function will not be checked when it is created. The validator function must take one argument of type oid, which will be the OID of the to-be-created function, and will typically return void.

A validator function would typically inspect the function body for syntactical correctness, but it can also look at other properties of the function, for example if the language cannot handle certain argument types. To signal an error, the validator function should use the ereport() function. The return value of the function is ignored.

Trusted

Specify that the call handler for the language is safe, that is, it does not offer an unprivileged user any functionality to bypass access restrictions. If this key word is omitted when registering the language, only users with the PostgreSQL superuser privilege can use this language to create new functions.

SQLite Objects

To start working with the server objects, you should create and open a connection.

You can attach a database, control-click the opened connection and choose **Attach Database** and enter the following information.

Option	Description
Database Name	Enter the database name which displays in Navicat.
Database File	Set the file path for a database.
Encrypted	Enable this option and provide **Password** when connecting to an encrypted SQLite database.

To detach a database, control-click it in the connection tree and choose **Detach Database**.

If you want to encrypt or decrypt a database, simply control-click it in the connection tree and choose **Encrypt Database** or **Decrypt Database**.

A special table named sqlite_master stores the complete database schema. To view the sqlite_master table, control-click the database and select **View Master Table** from the popup menu.

SQLite Tables

Relational databases use tables to store data. All operations on data are done on the tables themselves or produce another table as the result. A table is a set of rows and columns, and their intersections are fields. From a general perspective, columns within a table describe the name and type of data that will be found by row for that column's fields. Rows within a table represent records composed of fields that are described from left to right by their corresponding column's name and type. Each field in a row is implicitly correlated with each other field in that row. Click to open an object list for **Table**.

When open a table with graphical fields, control-click a table and select **Open Table (Quick)** from the popup menu. Faster performance for opening the graphical table, as BLOB fields (images) will not be loaded until you click on the cell. If you do wish Navicat loads all your images while opening the table, click **Open** button from the object list toolbar.

You can create a table shortcut by drag the table out. It provides a convenient way for you to open your table for entering data directly without activating the main Navicat.

To empty a table, control-click the selected table and choose **Empty Table** from the popup menu.

SQLite Table Fields

In the **Fields** tab, just simply click a field for editing. By using the field toolbar, you can create new, insert, move and drop the selected field. To search a field name, choose **Edit** -> **Find** -> **Find** or press CMD-F.

You can change the order of a field, simply drag and drop to field to desired location. To enable the drag and drop function, control-click the fields grid and choose **Drag to change table columns' order**.

Use the **Name** edit box to set the field name. Note that the name of the field must be unique among all the field names in the table.

The **Type** dropdown list defines the type (storage class) of the field data. See SQLite 2 Data Types and SQLite 3 Data Types for details.

Use the **Length** edit box to define the length of the field and use **Decimals** edit box to define the number of digits after the decimal point (the scale).

Not Null

Check this box to not allow the NULL values for the field.

Key

A Primary Key is a single field or combination of fields that uniquely defines a record. None of the fields that are part of the primary key can contain a null value.

Field's Properties

Note: The following options depend on the field type you are chosen.

Default Value

To set the default value for the field.

Collation

To specify the text collating function to use when comparing text entries for the column.

BINARY	Compare string data using memcmp(), regardless of text encoding.
NOCASE	The same as binary, except the 26 upper case characters of ASCII are folded to their lower case equivalents before the comparison is performed. Note that only ASCII characters are case folded. SQLite does not attempt to do full UTF case folding due to the size of the tables required.
RTRIM	The same as binary, except that trailing space characters are ignored.

Note: Support in SQLite 3.

Not null ON CONFLICT

To specify an algorithm used to resolve constraint conflicts if **Not Null** option is checked.

ROLLBACK	When a constraint violation occurs, an immediate ROLLBACK occurs, thus ending the current transaction, and the command aborts with a return code of SQLITE_CONSTRAINT. If no transaction is active (other than the implied transaction that is created on every command) then this algorithm works the same as ABORT.
ABORT	When a constraint violation occurs, the command backs out any prior changes it might have made and aborts with a return code of SQLITE_CONSTRAINT. But no ROLLBACK is executed so changes from prior commands within the same transaction are preserved. This is the default behavior.
FAIL	When a constraint violation occurs, the command aborts with a return code SQLITE_CONSTRAINT. But any changes to the database that the command made prior to encountering the constraint violation are preserved and

	are not backed out. For example, if an UPDATE statement encountered a constraint violation on the 100th row that it attempts to update, then the first 99 row changes are preserved but changes to rows 100 and beyond never occur.
IGNORE	When a constraint violation occurs, the one row that contains the constraint violation is not inserted or changed. But the command continues executing normally. Other rows before and after the row that contained the constraint violation continue to be inserted or updated normally. No error is returned when the IGNORE conflict resolution algorithm is used.
REPLACE	When a UNIQUE constraint violation occurs, the pre-existing rows that are causing the constraint violation are removed prior to inserting or updating the current row. Thus the insert or update always occurs. The command continues executing normally following REPLACE. No error is returned by the REPLACE conflict resolution. If a NOT NULL constraint violation occurs, the NULL value is replaced by the default value for that column. If the column has no default value, then the ABORT algorithm is used. If a CHECK constraint violation occurs then the IGNORE algorithm is used.

Auto Increment

The AUTO INCREMENT attribute can be used to generate a unique identity for new rows. To start with the AUTO INCREMENT value other than 1, you can set that value in Options tab.

SQLite Table Indexes

Index provides a faster access path to table data. It is created using one or more columns of a table to speed SQL statement execution on that table.

In the **Indexes** tab, just simply click an index field for editing. By using the index toolbar, you can create new, edit and delete the selected index field.

Use the **Name** edit box to set the index name.

To include field(s) in the index, click **Fields** to open the editor for editing.

Unique

All values of the indexed column(s) must only occur once.

Fields Editor

Select the field(s) from the **Field** list. You can also use the arrow buttons to change the index field(s) order.

Collate

To define a collating sequence used for text entries in that column. The default collating sequence is the collating sequence defined for that column.

BINARY	Compares string data using memcmp(), regardless of text encoding.
NOCASE	The same as binary, except the 26 upper case characters of ASCII are folded to their lower case equivalents before the comparison is performed. Note that only ASCII characters are case folded. SQLite does not

	attempt to do full UTF case folding due to the size of the tables required.
RTRIM	The same as binary, except that trailing space characters are ignored.

Note: Support in SQLite 3.

Order

To indicate sort order - ascending "ASC" or descending "DESC".

SQLite Table Foreign Keys

A foreign key is a field in a relational table that matches the primary key column of another table.

In the **Foreign Keys** tab, just simply click a foreign key field for editing. By using the foreign key toolbar, you can create new, edit and delete the selected foreign key field.

Use the **Name** edit box to enter a name for the new key.

Use the **Referenced Table** dropdown list to select a foreign table.

To include field(s)/referenced field(s) to the key, click **Fields** or **Referenced Fields** to open the editor(s) for editing.

The **On Delete** and **On Update** dropdown list define the type of the actions to be taken.

RESTRICT	The "RESTRICT" action means that the application is prohibited from deleting (for ON DELETE RESTRICT) or modifying (for ON UPDATE RESTRICT) a parent key when there exists one or more child keys mapped to it.
NO ACTION	Configuring "NO ACTION" means just that: when a parent key is modified or deleted from the database, no special action is taken.
CASCADE	A "CASCADE" action propagates the delete or update operation on the parent key to each dependent child key. For an "ON DELETE CASCADE" action, this means that each row in the child table that was associated with the deleted parent row is also deleted. For an "ON UPDATE CASCADE" action, it means that the values stored in each dependent child key are modified to match the new parent key values.
SET NULL	If the configured action is "SET NULL", then when a parent key is deleted (for ON DELETE SET NULL) or modified (for ON UPDATE SET NULL), the child key columns of all rows in the child table that mapped to the parent key are set to contain SQL NULL values.
SET DEFAULT	The "SET DEFAULT" actions are similar to "SET NULL", except that each of the child key columns is set to contain the columns default value instead of NULL.

Deferred

Deferred foreign key constraints are not checked until the transaction tries to COMMIT.

Related topic:

Foreign Keys Data Selection

SQLite Table Uniques

Unique constraints ensure that the data contained in a column or a group of columns is unique with respect to all the rows in the table.

In the **Uniques** tab, just simply click an unique field for editing. By using the unique toolbar, you can create new, edit and delete the selected unique field.

Use the **Name** edit box to set the unique name.

To set field(s) as unique, click **Fields** to open the editor(s) for editing.

ON CONFLICT

To specify an algorithm used to resolve constraint conflicts.

ROLLBACK	When a constraint violation occurs, an immediate ROLLBACK occurs, thus ending the current transaction, and the command aborts with a return code of SQLITE_CONSTRAINT. If no transaction is active (other than the implied transaction that is created on every command) then this algorithm works the same as ABORT.
ABORT	When a constraint violation occurs, the command backs out any prior changes it might have made and aborts with a return code of SQLITE_CONSTRAINT. But no ROLLBACK is executed so changes from prior commands within the same transaction are preserved. This is the default behavior.
FAIL	When a constraint violation occurs, the command aborts with a return code SQLITE_CONSTRAINT. But any changes to the database that the command made prior to encountering the constraint violation are preserved and are not backed out. For example, if an UPDATE statement encountered a constraint violation on the 100th row that it attempts to update, then the first 99 row changes are preserved but changes to rows 100 and beyond never occur.
IGNORE	When a constraint violation occurs, the one row that contains the constraint violation is not inserted or changed. But the command continues executing normally. Other rows before and after the row that contained the constraint violation continue to be inserted or updated normally. No error is returned when the IGNORE conflict resolution algorithm is used.
REPLACE	When a UNIQUE constraint violation occurs, the pre-existing rows that are causing the constraint violation are removed prior to inserting or updating the current row. Thus the insert or update always occurs. The command continues executing normally following REPLACE. No error is returned by the REPLACE conflict resolution. If a NOT NULL constraint violation occurs, the NULL value is replaced by the default value for that column. If the column has no default value, then the ABORT algorithm is used. If a CHECK constraint violation occurs then the IGNORE algorithm is used.

Fields Editor

Select the field(s) from the **Field** list. To remove the fields from the unique, uncheck them in the same way. You can also use the arrow buttons to change the unique field(s) order.

Collate

To define a collating sequence used for text entries in that column. The default collating sequence is the collating sequence defined for that column.

BINARY	Compares string data using memcmp(), regardless of text encoding.
NOCASE	The same as binary, except the 26 upper case characters of ASCII are folded to their lower case equivalents before the comparison is performed. Note that only ASCII characters are case folded. SQLite does not attempt to do full UTF case folding due to the size of the tables required.
RTRIM	The same as binary, except that trailing space characters are ignored.

Note: Support in SQLite 3.

Order

To indicate sort order - ascending "ASC" or descending "DESC".

SQLite Table Checks

A check constraint allows you to specify that the value in a certain column must satisfy a Boolean (truth-value) expression.

In the **Checks** tab, just simply click a check field for editing. By using the check toolbar, you can create new, edit and delete the selected check field.

Note: Checks are supported from SQLite version 3.3.0 or later.

Use the **Name** edit box to set the check name.

Expression

Set the condition for checking, e.g. "field_name1 > 0 AND field_name2 > field_name1" in the **Expression** edit box.

SQLite Table Triggers

A trigger is a database operation that is automatically performed when a specified database event occurs.

In the **Triggers** tab, just simply click a trigger field for editing. By using the trigger toolbar, you can create new, edit and delete the selected trigger field.

Name

Set the trigger name.

Fires

Determine when the trigger actions will be executed relative to the insertion, modification or removal of the associated row.

Events

It indicates the kind of statement that activates the trigger.

Insert	Fire the trigger whenever an INSERT statement adds a row to a table.
Update	Fire the trigger whenever an UPDATE statement changes a value in one of the columns specified in **Update Of**. If no **Update Of** are present, the trigger will be fired whenever an UPDATE statement changes a value in any column of the table.
Delete	Fire the trigger whenever a DELETE statement removes a row from the table.

Update Of

Specify the fields for UPDATE statement trigger upon necessary.

Body

Type in the definition for the trigger.

When

Specify the trigger condition, which is a SQL condition that must be satisfied for the database to fire the trigger.

SQLite Table Options

Primary Key ON CONFLICT

To specify an algorithm used to resolve primary key constraint conflicts.

ROLLBACK	When a constraint violation occurs, an immediate ROLLBACK occurs, thus ending the current transaction, and the command aborts with a return code of SQLITE_CONSTRAINT. If no transaction is active (other than the implied transaction that is created on every command) then this algorithm works the same as ABORT.
ABORT	When a constraint violation occurs, the command backs out any prior changes it might have made and aborts with a return code of SQLITE_CONSTRAINT. But no ROLLBACK is executed so changes from prior commands within the same transaction are preserved. This is the default behavior.
FAIL	When a constraint violation occurs, the command aborts with a return code SQLITE_CONSTRAINT. But any changes to the database that the command made prior to encountering the constraint violation are preserved and are not backed out. For example, if an UPDATE statement encountered a constraint violation on the 100th row that it attempts to update, then the first 99 row changes are preserved but changes to rows 100 and beyond never occur.
IGNORE	When a constraint violation occurs, the one row that contains the constraint violation is not inserted or changed. But the command continues executing normally. Other rows before and after the row that contained the constraint violation continue to be inserted or updated normally. No error is returned when the IGNORE conflict resolution algorithm is used.
REPLACE	When a UNIQUE constraint violation occurs, the pre-existing rows that are causing the constraint violation are removed prior to inserting or updating the current row. Thus the insert or update always occurs. The command continues executing normally following REPLACE. No error is returned by the REPLACE conflict resolution. If a NOT NULL constraint violation occurs, the NULL value is replaced by the default value for that column. If the column has no default value, then the ABORT algorithm is used. If a CHECK constraint violation occurs then the IGNORE algorithm is used.

Auto Increment

Set/Reset the **Auto Increment** value in the edit field. The auto increment value indicates the value for next record.

SQLite Views

Views are useful for allowing users to access a set of tables as if it were a single table, and limiting their access to just that. Views can also be used to restrict access to rows (a subset of a particular table). Click to open an object list for **View**.

You can create a view shortcut by drag the view out. It provides a convenient way for you to open your view without activating the main Navicat.

Button	Description
Preview	Preview and/or Explain the view.

Note: You can choose to show the Result tab below the editor or in a new tab by selecting **Edit** -> **Show Result** -> **Below Query Editor** or **In a New Tab**.

View Builder (Available only in Full Version)

View Builder allows you to build views visually. It allows you to create and edit views without knowledge of SQL. See Query Builder for details.

View Editor

You can edit the view definition as SQL statement (SELECT statement it implements).

Hint: To customize the view of the editor and find out more features for sql editing, see Editor Advanced Features.

View Viewer

View Viewer displays the view data as a grid. Data can be displayed in two modes: **Grid View** and **Form View**. See Table Viewer for details.

SQLite Indexes

Index provides a faster access path to table data. It is created using one or more columns of a table to speed SQL statement execution on that table. Click to open an object list for **Index**.

General Properties

Table Name

The table that contains the index.

Unique

An unique index indicates that no two rows of a table have duplicate values in the key columns.

Field

To define the field.

Collate

To define a collating sequence used for text entries in that column. The default collating sequence is the collating sequence defined for that column.

BINARY	Compare string data using memcmp(), regardless of text encoding.
NOCASE	The same as binary, except the 26 upper case characters of ASCII are folded to their lower case equivalents before the comparison is performed. Note that only ASCII characters are case folded. SQLite does not attempt to do full UTF case folding due to the size of the tables required.
RTRIM	The same as binary, except that trailing space characters are ignored.

Note: Support in SQLite 3.

Order

To indicate sort order - ascending "ASC" or descending "DESC".

SQLite Triggers

Triggers are database operations that are automatically performed when a specified database event occurs. Click to open an object list for **Trigger**.

See Triggers for details.

General Properties

Trigger Type

Define the trigger type: TABLE or VIEW.

Table Name or **View Name**

Choose a table or view.

Fire

Determine when the trigger actions will be executed relative to the insertion, modification or removal of the associated row.

Event

It indicates the kind of statement that activates the trigger.

INSERT	The trigger is activated whenever adding a row to a table.
DELETE	The trigger is activated whenever removing a row from the table.
UPDATE	The trigger is activated whenever changing a value in one of the fields selected in **Columns**.

When

Specify the trigger condition for the database to fire the trigger.

Columns

Specify the fields for UPDATE statement trigger upon necessary.

Definition

You can edit valid SQL statements in the trigger definition inside *BEGIN* and *END*.

SQL Server Objects

To start working with the server objects, you should create and open a connection. If the server is empty, you need to control-click the connection in the connection tree and choose **New Database** to create a new database.

Note: SQL Azure does not support **Comment** tab.

General Properties for SQL Azure

Database Name

Define the name of the database.

Edition

Choose the edition of the database: web or business.

Max Size

Choose the maximum size of the database.

General Properties for SQL Server

Database Name

Define the name of the database.

Owner

Choose the owner of the database.

Collation

Choose the default collation for the database. Collation name can be either a Windows collation name or a SQL collation name. If not specified, the database is assigned the default collation of the instance of SQL Server. A collation name cannot be specified on a database snapshot.

Recovery Model

Control database recovery options and disk I/O error checking.

FULL	Provide full recovery after media failure by using transaction log backups. If a data file is

	damaged, media recovery can restore all committed transactions.
BULK_LOGGED	Provide recovery after media failure by combining the best performance and least amount of log-space use for certain large-scale or bulk operations.
SIMPLE	A simple backup strategy that uses minimal log space is provided. Log space can be automatically reused when it is no longer required for server failure recovery.

Compatibility Level

Choose the version of SQL Server with which the database is to be made compatible.

Filegroups Properties for SQL Server

Rows

Add or delete a filegroup. PRIMARY filegroup cannot be deleted.

Filestream

Add or delete a FILESTREAM filegroup.

Note: Support from SQL Server 2008 or later.

Files Properties for SQL Server

Name

Specify the logical name for the file.

File Type

Choose the file type.

Directory

The path used by the operating system when you create the file.

File Name

The file name used by the operating system when you create the file.

Filegroup

Choose the filegroup.

Initial Size

Specify the size of the file.

Unlimited File Size

Specify that the file grows until the disk is full. In SQL Server, a log file specified with unlimited growth has a maximum size of 2 TB, and a data file has a maximum size of 16 TB.

Max Size

Specify the maximum size to which the file can grow.

Enable Auto Growth

Check this option if you want to allow automatic growth.

Growth

Specify the automatic growth increment of the file.

Advanced Properties (Automatic) for SQL Server

Auto Close

If this option is on, the database is shut down cleanly and its resources are freed after the last user exits.

Auto Create Statistics

If this option is on, the query optimizer creates statistics on single columns in query predicates, as necessary, to improve query plans and query performance.

Auto Shrink

If this option is on, the database files are candidates for periodic shrinking.

Auto Update Statistics

Specify that the query optimizer updates statistics when they are used by a query and when they might be out-of-date.

Auto Update Statistics Asynchronously

Specify that statistics updates for the AUTO_UPDATE_STATISTICS option are asynchronous. The query optimizer does not wait for statistics updates to complete before it compiles queries.

Note: Support from SQL Server 2005 or later.

Advanced Properties (Change Tracking) for SQL Server

Note: Support from SQL Server 2008 or later.

Change Tracking

Enable change tracking for the database.

Retention Period

Specify the minimum period for keeping change tracking information in the database. Data is removed only when **Auto Cleanup** is checked.

Auto Cleanup

Change tracking information is automatically removed after the specified retention period.

Advanced Properties (Cursor) for SQL Server

Close Cursor On Commit Enabled

If this option is on, any cursors open when a transaction is committed or rolled back are closed.

Default Cursor

LOCAL	When LOCAL is specified and a cursor is not defined as GLOBAL when created, the scope of the cursor is local to the batch, stored procedure, or trigger in which the cursor was created. The cursor name is valid only within this scope. The cursor can be referenced by local cursor variables in the batch, stored procedure, or trigger, or a stored procedure OUTPUT parameter. The cursor is implicitly deallocated when the batch, stored procedure, or trigger ends, unless it was passed back in an OUTPUT parameter. If the cursor is passed back in an OUTPUT parameter, the cursor is deallocated when the last variable that references it is deallocated or goes out of scope.
GLOBAL	When GLOBAL is specified, and a cursor is not defined as LOCAL when created, the scope of the cursor is global to the connection. The cursor name can be referenced in any stored procedure or batch executed by the connection.

Advanced Properties (Recovery) for SQL Server

Page Verify

Discover damaged database pages caused by disk I/O path errors. Disk I/O path errors can be the cause of database corruption problems and are generally caused by power failures or disk hardware failures that occur at the time the page is being written to disk.

NONE	Database page writes will not generate a CHECKSUM or TORN_PAGE_DETECTION value. SQL Server will not verify a checksum or torn page during a read even if a CHECKSUM or TORN_PAGE_DETECTION value is present in the page header.
TORN_PAGE_DETECTION	Save a specific 2-bit pattern for each 512-byte sector in the 8-kilobyte (KB) database page and stored in the database page header when the page is written to disk.
CHECKSUM	Calculate a checksum over the contents of the whole page and stores the value in the page header when a page is written to disk.

Advanced Properties (Service Broker) for SQL Server

Note: Support from SQL Server 2005 or later.

Broker Enabled

Specify that Service Broker is enabled for the specified database. Message delivery is started, and the is_broker_enabled flag is set to true in the sys.databases catalog view. The database retains the existing Service Broker identifier.

Honor Broker Priority

Send operations take into consideration the priority levels that are assigned to conversations. Messages from conversations that have high priority levels are sent before messages from conversations that are assigned low priority levels.

Note: Support from SQL Server 2008 or later.

Advanced Properties (SQL) for SQL Server

ANSI Null Default

Check this option if you want to determines the default value as NULL.

ANSI Nulls Enabled

If this option is on, all comparisons to a null value evaluate to UNKNOWN.

ANSI Padding Enabled

If this option is on, strings are padded to the same length before conversion or inserting to a varchar or nvarchar data type.

ANSI Warnings Enabled

If this option is on, errors or warnings are issued when conditions such as divide-by-zero occur or null values appear in aggregate functions.

Arithmetic Abort Enabled

If this option is on, a query is ended when an overflow or divide-by-zero error occurs during query execution.

Concatenate Null Yields Null

If this option is on, the result of a concatenation operation is NULL when either operand is NULL.

Numeric Round-Abort

If this option is on, an error is generated when loss of precision occurs in an expression.

Quoted Identifiers Enabled

If this option is on, double quotation marks can be used to enclose delimited identifiers.

Recursive Triggers Enabled

If this option is on, Recursive firing of AFTER triggers is allowed.

Advanced Properties (State) for SQL Server

Database Read-Only

If this option is on, users can read data from the database but not modify it.

Database State

Choose the state of the database.

OFFLINE	The database is closed, shut down cleanly, and marked offline. The database cannot be modified while it is offline.
ONLINE	The database is open and available for use.
EMERGENCY	The database is marked READ_ONLY, logging is disabled, and access is limited to members of the sysadmin fixed server role. EMERGENCY is primarily used for troubleshooting purposes.

Restrict Access

Control user access to the database.

SINGLE_USER	Specifies that only one user at a time can access the database.
RESTRICTED_USER	RESTRICTED_USER allows for only members of the db_owner fixed database role and dbcreator and sysadmin fixed server roles to connect to the database, but does not limit their number.

MULTI_USER	All users that have the appropriate permissions to connect to the database are allowed.

Encryption Enabled

Check this option if you want to encrypt the database.

Note: Support from SQL Server 2008 or later.

Advanced Properties (Miscellaneous) for SQL Server

Note: Support from SQL Server 2005 or later.

Cross-database Ownership Chaining Enabled

If this option is on, database can be the source or target of a cross-database ownership chain.

Trustworthy

If this option is on, database modules (for example, user-defined functions or stored procedures) that use an impersonation context can access resources outside the database.

Date Correlation Optimization Enabled

SQL Server maintains correlation statistics between any two tables in the database that are linked by a FOREIGN KEY constraint and have datetime columns.

Parameterization

SIMPLE	Queries are parameterized based on the default behavior of the database.
FORCED	SQL Server parameterizes all queries in the database.

VarDecimal Storage Format Enabled

Indicate that decimal and numeric data types are stored by using the vardecimal storage format.

SQL Server Backup/Restore (Available only in Full Version)

You can backup and restore your SQL Server databases. The SQL Server backup and restore component provides an important safeguard for protecting critical data. Click to open an object list for **SQL Server Backup**.

Note: The backup file is stored in the server.

SQL Server Backup

Before starting the backup process, click **Generate SQL** button to review the SQL. Then, you can click the **Start** button to run it. If you want to backup with the setting of an existing backup file, you can control-click a backup file in the object list and choose **Backup from this Setting**.

General Properties

Backup Type

Full	Perform a full backup.
Differential	Specify that the database or file backup should consist only of the portions of the database or file changed since the last full backup.
Transaction Log	Specify a backup of the transaction log only. Note: Support from SQL Server 2012 or later and SQL Azure.

Backup Name

Specify the name of the backup set.

Description

Specify the free-form text describing the backup set.

Password

Set the password for the backup set.

New media set

Create a new media set for the backup. To add a Backup Device or file to the list, click ✚button.

Existing media set

Choose an existing **Media set** for the backup.

Components

Choose the **Backup Components**: **Database**, **Partial** or **Files and Filegroups**.

Note: Files and Filegroups option is supported from SQL Server 2012 or later and SQL Azure.

Options

Copy only

Specify that the backup is a copy-only backup, which does not affect the normal sequence of backups.

Never expire

Specify the backup set never expire.

Expire after ☐ **days**

Specify the number of days that must elapse before this backup media set can be overwritten.

Expire on

Specify when the backup set expires and can be overwritten.

New Media Set Name *(only for New media set)*

The name of the new media set.

New Media Set Description *(only for New media set)*

Specify the free-form text description of the media set.

Use password *(only for New media set)*

Set the password for the media set.

Append to the existing backup set *(only for Existing media set)*

Indicate that the backup set is appended to the specified media set, preserving existing backup sets.

Overwrite all existing backup sets *(only for Existing media set)*

Specify that all backup sets should be overwritten, but preserves the media header.

Check media set name and backup set expiration *(only for Existing media set)*

Control whether a backup operation checks the expiration date and time of the backup sets on the media before overwriting them.

Media Set Name *(only for Existing media set)*

Specify the media name for the entire backup media set.

Media set is password protected *(only for Existing media set)*

Enter the password for the media set.

Verify backup when finished

Check this options to verify the backup.

Perform checksum before writing to media

Enable the backup checksums.

Continue on error

Instruct backup to continue despite encountering errors such as invalid checksums or torn pages.

Truncate the transaction log *(only for Transaction Log backup)*

Check this option to truncate the transaction log.

Back up the tail of the log and leave the database in the restoring state *(only for Transaction Log backup)*

Choose this option to back up the tail of the log and leaves the database in the RESTORING state.

Compress Backup

Specify whether backup compression is performed on this backup.

Note: Support from SQL Server 2008 or later and SQL Azure.

SQL Server Restore

Before starting the restore process, click **Generate SQL** button to review the SQL. Then, you can click the **Start** button to run it. If you want to restore from a file that is not listed in the object list, you can control-click anywhere in object list and choose **Restore from File**.

General Properties for Restore Backup

Restore to Database

Select the database to restore.

Latest possible *(only for Transaction Log backup)*

Choose this option if do not have the restore point.

Specific time *(only for Transaction Log backup)*

Specify that the database be restored to the state it was in as of the date and time.

Marked transaction *(only for Transaction Log backup)*

Specify recovery to a specified recovery point.

Include marked transaction *(only for Transaction Log backup)*

The specified transaction is included in the recovery, but it is committed only if it was originally committed when the transaction was actually generated.

General Properties for Restore from File

Restore to Database

Select the database to restore.

Source of Backup Set

To add a Backup Device or file to the list, click + button.

Latest possible

Choose this option if do not have the restore point.

Specific time

Specify that the database be restored to the state it was in as of the date and time.

Restore Plan

Select the database backup files in the list.

Advanced Properties

Restore database files to

Specify that the data or log file should be moved by restoring it to the location specified in **Restore To**.

WITH REPLACE

Specify that SQL Server should create the specified database and its related files even if another database already exists with the same name.

WITH RESTRICTED_USER

Restrict access for the newly restored database to members of the db_owner, dbcreator, or sysadmin roles.

WITH KEEP_REPLICATION

KEEP_REPLICATION should used when setting up replication to work with log shipping.

RECOVERY

Roll back all uncommitted transactions. After the recovery process, the database is ready for use.

NORECOVERY

Do not roll back the uncommitted transactions.

STANDBY

Specify a **Standby File** that allows the recovery effects to be undone.

SQL Server Schemas

A schema contains named objects (tables, views, functions, etc) whose names may duplicate those of other objects existing in other schemas. The schema name must be distinct from any existing schema name in the current database.

Note: SQL Server 2000 or below does not support edit, create and delete schema.

SQL Azure does not support the **Comment** tab.

General Properties

Schema Name

The name of a schema which is identified within the database.

Owner

The name of the database-level principal that will own the schema. This principal may own other schemas, and may not use the current schema as its default schema.

SQL Server Tables

Tables are database objects that contain all the data in a database. A table definition is a collection of columns. In tables, data is organized in a row-and-column format similar to a spreadsheet. Each row represents a unique record, and each column represents a field within the record. Click to open an object list for **Table**.

When open a table with graphical fields, control-click a table and select **Open Table (Quick)** from the popup menu. Faster performance for opening the graphical table, as BLOB fields (images) will not be loaded until you click on the cell. If you do wish Navicat loads all your images while opening the table, click **Open** button from the object list toolbar.

You can create a table shortcut by drag the table out. It provides a convenient way for you to open your table for entering data directly without activating the main Navicat.

To empty a table, control-click the selected table and choose **Empty Table** from the popup menu.

Note: SQL Azure does not support the **Storage** and **Comment** tab.

SQL Server Table Fields

In the **Fields** tab, just simply click a field for editing. By using the field toolbar, you can create new and drop the selected field. To search a field name, choose **Edit** -> **Find** -> **Find** or press CMD-F.

Use the **Name** edit box to set the field name. Note that the name of the field must be unique among all the field names in the table.

The **Type** dropdown list defines the type of the field data. See SQL Server Data Type and SQL Azure Support Data Type for details.

Use the **Size** edit box to define the length of the field and use **Scale** edit box to define the number of digits after the decimal point (the scale) for Floating Point data type.

Note: Be careful when shortening the field length as losing data might be caused.

Not Null

Check this box to not allow the NULL values for the field.

Key

A Primary Key is a single field or combination of fields that uniquely defines a record. None of the fields that are part of the primary key can contain a null value.

Field's Properties

Note: The following options depend on the field type you are chosen.

Identity

Indicate that the new column is an identity column.

Row GUID Column

Indicate that the new column is a row GUID column. Only one uniqueidentifier column per table can be designated as the ROWGUIDCOL column.

Note: SQL Azure does not support.

Collate

Specify the collation for the column.

Column Set

Combine all of the sparse columns of a table into a structured output.

Note: Support from SQL Server 2008 or later.

Filestream

Specify FILESTREAM storage for the varbinary(max) BLOB data.

Note: Support from SQL Server 2008 or later.

Schema

Set the schema of the user defined type.

User-defined Type

Set the user defined type.

Expression

Set an expression that defines the value of a computed column.

Persisted

Specify that the SQL Server Database Engine will physically store the computed values in the table, and update the values when any other columns on which the computed column depends are updated.

Default Value

Set the default value for the field.

Sparse

Indicate that the column is a sparse column.

Note: Support from SQL Server 2008 or later.

Comment

Set any optional text describing the current field.

Note: SQL Azure does not support.

SQL Server Table Indexes

Indexes are optional structures associated with tables. You can create indexes on one or more columns of a table to speed SQL statement execution on that table.

In the **Indexes** tab, just simply click an index field for editing. By using the index toolbar, you can create new, edit and delete the selected index field.

Use the **Name** edit box to set the index name.

To include field(s) in the index, click **Fields** to open the editor for editing.

Note: Some of data types do not allow indexing. For example: text

The **Index Type** dropdown list defines the type of the table index.

Clustered	Create an index in which the logical order of the key values determines the physical order of the corresponding rows in a table.
Nonclustered	Create an index that specifies the logical ordering of a table. With a nonclustered index, the physical order of the data rows is independent of their indexed order.
Spatial	Create a spatial index on a specified table and column. An index can be created before there is data in the table. Note: Support from SQL Server 2008 or later and SQL Azure.
XML	Create an XML index on a specified table. An index can be created before there is data in the table. Note: Support from SQL Server 2005 or later.

Unique

Create a unique index on a table.

Comment

Specify the comment of the index.

Note: SQL Azure does not support.

SQL Server Table Foreign Keys

A foreign key is a field in a relational table that matches the primary key column of another table. The foreign key can be used to cross-reference tables.

In the **Foreign Keys** tab, just simply click a foreign key field for editing. By using the foreign key toolbar, you can create new, edit and delete the selected foreign key field.

Use the **Name** edit box to enter a name for the new key.

Use the **Referenced Schema** and **Referenced Table** dropdown lists to select a foreign schema and table respectively.

To include field(s)/referenced field(s) to the key, click **Fields** or **Referenced Fields** to open the editor(s) for editing.

The **On Delete** and **On Update** dropdown list define the type of the actions to be taken.

No Action	The Database Engine raises an error and the delete or update action on the row in the parent table is rolled back.
Cascade	Corresponding rows are deleted from or updated in the referencing table if that row is deleted from or updated in the parent table.
Set Null	All the values that make up the foreign key are set to NULL when the corresponding row in the parent table is deleted or updated.
Set Default	All the values that make up the foreign key are set to their default values when the corresponding row in the parent table is deleted or updated.

Not for Replication

The constraint is not enforced when replication agents perform insert, update, or delete operations.

Note: SQL Azure does not support.

Enable

You can choose whether to enable/disable the foreign key constraint by checking/unchecking the box.

Comment

Specify the comment of the foreign key.

Note: SQL Azure does not support.

Related topic:

Foreign Keys Data Selection

SQL Server Table Uniques

Unique constraints ensure that the data contained in a column or a group of columns is unique with respect to all the rows in the table.

In the **Uniques** tab, just simply click an unique field for editing. By using the unique toolbar, you can create new, edit and delete the selected unique field.

Use the **Name** edit box to set the unique name.

To set field(s) as unique, click **Fields** to open the editor for editing.

Unique Type

Indicate that a clustered or nonclustered index is created for the unique constraint.

Comment

Specify the comment of the unique.

Note: SQL Azure does not support.

SQL Server Table Checks

A check a constraint that enforces domain integrity by limiting the possible values that can be entered into a column or columns.

In the **Checks** tab, just simply click a check field for editing. By using the check toolbar, you can create new, edit and delete the selected check field.

Use the **Name** edit box to set the check name.

Expression

Set the condition for checking, e.g. "field_name1 > 0 AND field_name2 > field_name1" in the **Expression** edit box.

Not for Replication

The constraint is not enforced when replication agents perform insert, update, or delete operations.

Note: SQL Azure does not support.

Enable

You can choose whether to enable/disable the check constraint by checking/unchecking the box.

Comment

Specify the comment of the check.

Note: SQL Azure does not support.

SQL Server Table Triggers

A trigger is a special kind of stored procedure that automatically executes when an event occurs in the database server.

In the **Triggers** tab, just simply click a trigger field for editing. By using the trigger toolbar, you can create new, edit and delete the selected trigger field.

Use the **Name** edit box to set the trigger name.

Use the **Fires** dropdown list to define the trigger action time.

After	Specify that the DML trigger is fired only when all operations specified in the triggering SQL statement have executed successfully.
Instead of	Specify that the DML trigger is executed instead of the triggering SQL statement, therefore, overriding the actions of the triggering statements.

Delete

The trigger is activated whenever a row is deleted from the table.

Insert

The trigger is activated whenever a new row is inserted into the table.

Update

The trigger is activated whenever a row is modified.

Enable

You can choose whether to enable/disable the trigger by checking/unchecking the box.

The **Body** tab defines the statement to execute when the trigger activates. To include your statement, just simply click to write. If you want to execute multiple statements, use the **BEGIN ... END** compound statement construct.

Execute as

Specify the security context under which the trigger is executed.

Note: Support from SQL Server 2005 or later and SQL Azure.

User

Choose a user that the trigger executes in.

Note: Support from SQL Server 2005 or later and SQL Azure.

Definition Type

Choose the type of definition.

Note: Support from SQL Server 2005 or later.

Encrypted

Obfuscate the text of the CREATE TRIGGER statement.

Note: Support from SQL Server 2005 or later.

Not for replication

Indicate that the trigger should not be executed when a replication agent modifies the table that is involved in the trigger.

Note: SQL Azure does not support.

Append

Specify that an additional trigger of an existing type should be added.

Note: SQL Azure does not support.

Comment

Specify the comment of the trigger.

Note: SQL Azure does not support.

SQL Server Table Storage

Store on filegroup

Option	Description
Filegroup	Choose a filegroup that storing the table.
Text/Image Filegroup	Choose a filegroup for storing text, ntext, image, xml, varchar(max), nvarchar(max),

	varbinary(max), and CLR user-defined type columns.
Filestream Filegroup	Choose a filegroup for FILESTREAM data. Note: Support from SQL Server 2008 or later.

Store on partition scheme

Option	Description
Partition Scheme	Choose a partition scheme that storing the table.
Partition Column	Choose a partition column name.
Filestream Partition Scheme	Choose a partition scheme for FILESTREAM data. Note: Support from SQL Server 2008 or later.
Data Compression	Specify the data compression option for the specified table, partition number, or range of partitions. Note: Support from SQL Server 2008 or later.

Note: Support from SQL Server 2005 or later.

SQL Server Table Options

Seed

The value used for the very first row loaded into the table.

Increment

The incremental value added to the identity value of the previous row loaded.

Current

Set the current identity value.

Note: SQL Azure does not support.

Not for Replication

Values are not incremented in identity columns when replication agents perform inserts.

Note: SQL Azure does not support.

Change Tracking

Specify change tracking is enabled for the table.

Note: Support from SQL Server 2008 or later.

Track Columns Updated

Specify the Database Engine tracks which change tracked columns were updated.

Note: Support from SQL Server 2008 or later.

Lock Escalation

Specify the allowed methods of lock escalation for a table.

Note: Support from SQL Server 2008 or later.

SQL Server Views

A view can be thought of as either a virtual table or a stored query. Unless a view is indexed, its data is not stored in the database as a distinct object. What is stored in the database is a SELECT statement. The result set of the SELECT statement forms the virtual table returned by the view. A user can use this virtual table by referencing the view name in Transact-SQL statements the same way a table is referenced. Click to open an object list for **View**.

You can create a view shortcut by drag the view out. It provides a convenient way for you to open your view without activating the main Navicat.

Note: SQL Azure does not support the **Comment** tab.

Button	Description
Preview	Preview and/or Explain the view.

Note: You can choose to show the Result tab below the editor or in a new tab by selecting **Edit** -> **Show Result** -> **Below Query Editor** or **In a New Tab**.

View Builder (Available only in Full Version)

View Builder allows you to build views visually. It allows you to create and edit views without knowledge of SQL. See Query Builder for details.

View Editor

You can edit the view definition as SQL statement (SELECT statement it implements).

Hint: To customize the view of the editor and find out more features for sql editing, see Editor Advanced Features.

Advanced Properties

Encrypted

Encrypt the entries in sys.syscomments that contain the text of the CREATE VIEW statement.

Note: SQL Azure does not support.

Schema binding

Bind the view to the schema of the underlying table or tables.

View metadata

Specify that the instance of SQL Server will return to the DB-Library, ODBC, and OLE DB APIs the metadata information about the view, instead of the base table or tables, when browse-mode metadata is being requested for a query that references the view.

Check option

Force all data modification statements executed against the view to follow the criteria set within select_statement.

View Viewer

View Viewer displays the view data as a grid. Data can be displayed in two modes: **Grid View** and **Form View**. See Table Viewer for details.

SQL Server Functions/Procedures

A user-defined function, which is a Transact-SQL or common language runtime (CLR) routine that accepts parameters, performs an action, such as a complex calculation, and returns the result of that action as a value. The return value can either be a scalar (single) value or a table. Click f_0 to open an object list for **Function**.

Stored procedures are similar to procedures in other programming languages in that they can:

- Accept input parameters and return multiple values in the form of output parameters to the calling procedure or batch.
- Contain programming statements that perform operations in the database, including calling other procedures.
- Return a status value to a calling procedure or batch to indicate success or failure (and the reason for failure).

Note: SQL Azure does not support the **Comment** tab.

Function Pop-up Window

Click the **+Add** from the object list toolbar. A Window will pop-up and it allows you to create a procedure/function easily.

Type

Define whether it is a procedure/function.

Name

Specify the name for the procedure/function.

Define the parameter(s) for the procedure/function. Set the parameter **Name**, **Schema Type**, **Type**, **Default Value**, **Output** and **Read Only** under corresponding columns.

Definition

The **Code Outline** window displays information about the function/procedure including parameter, code body, etc. To show the **Code Outline** window, simply choose **Edit** -> **Show Code Outline** from main menu.

Note: Available only in Full Version.

Button	Description
	Refresh the code outline.
	Turn mouse over highlight on or off.
	Show the code outline in a detail way.
	Sort by type and name.
	Expand the selected item.
	Collapse the selected item.

Hint: To customize the view of the editor and find out more features for sql editing, see Editor Advanced Features.

Result

To run the procedure/function, click **Execute** on the toolbar. If the SQL statement is correct, the statement will be executed and, if the statement is supposed to return data, the **Result** tab opens with the data returned by the procedure/function. If an error occurs while executing the procedure/function, execution stops, the appropriate error message is displayed. If the function/procedure requires input parameter, the **Input Parameters** box will popup.

SQL Server Indexes

An index in a database lets you quickly find specific information in a table or indexed view. An index contains keys built from one or more columns in the table, or view, and pointers that map to the storage location of the specified data. You can significantly improve the performance of database queries and applications by creating well-designed indexes to support your queries. Indexes can reduce the amount of data that must be read to return the query result set. Indexes can also enforce uniqueness on the rows in a table, ensuring the data integrity of the table data. Click -> **Index** to open an object list for **Index**.

Note: SQL Azure does not support the **Comment** tab.

You can choose the index **Type**: Nonclustered, Clustered, XML or Spatial.

Note: XML index is supported from SQL Server 2005 or later.
Spatial index is supported from SQL Server 2008 or later and SQL Azure.

General Properties for Nonclustered and Clustered Indexes

Unique

A unique index is one in which no two rows are permitted to have the same index key value.

Table/View

Choose to create a table index or a view index.

Table Name or **View Name**

Select a table or a view.

Fields

Select the column or columns on which the index is based and the sorting order.

Included Columns *(only for Nonclustered Index)*

Select the non-key columns to be added to the leaf level of the nonclustered index.

Note: Support from SQL Server 2005 or later and SQL Azure.

Filter Properties for Nonclustered Index

To create a filtered index, specify which rows to include in the index.

Note: Support from SQL Server 2008 or later and SQL Azure.

Advanced Properties for Nonclustered and Clustered Indexes

Ignore duplicate key values

A warning message will occur when duplicate key values are inserted into a unique index. Only the rows violating the uniqueness constraint will fail.

Recompute statistics

Enable automatic statistics updating.

Allow row locks

Row locks are allowed when accessing the index. The Database Engine determines when row locks are used.

Note: Support from SQL Server 2005 or later.

Allow page locks

Page locks are allowed when accessing the index. The Database Engine determines when page locks are used.

Note: Support from SQL Server 2005 or later.

Fill Factor (%)

Specify a percentage that indicates how full the Database Engine should make the leaf level of each index page during index creation or rebuild.

Note: SQL Azure does not support.

Pad index

The percentage of free space that is specified by fillfactor is applied to the intermediate-level pages of the index.

Note: Support from SQL Server 2005 or later.

Sort in tempdb

Specify to store temporary sort results in tempdb.

Note: SQL Azure does not support.

Online

Long-term table locks are not held for the duration of the index operation.

Note: Support from SQL Server 2005 or later and SQL Azure.

Max. degree of parallelism

Override the max degree of parallelism configuration option for the duration of the index operation.

Note: Support from SQL Server 2005 or later.

Storage Properties for Nonclustered and Clustered Indexes

Note: SQL Azure does not support this tab.

Store on filegroup

Option	Description
Filegroup	Choose a filegroup that storing the index.
Filestream Filegroup	Choose a filegroup for FILESTREAM data. Note: Support from SQL Server 2008 or later.

Store on partition scheme

Option	Description
Partition Scheme	Choose a partition scheme that storing the index.
Partition Column	Choose a partition column name.
Filestream Partition Scheme	Choose a partition scheme for FILESTREAM data. Note: Support from SQL Server 2008 or later.
Data Compression	Specify the data compression option for the specified index, partition number, or range of partitions. Note: Support from SQL Server 2008 or later.

Note: Support from SQL Server 2005 or later.

General Properties for XML Index

Table/View

Must be TABLE.

Table Name

Select a table.

XML Column

Select the xml column on which the index is based.

XML Index Type

PRIMARY	A clustered index is created with the clustered key formed from the clustering key of the user table and an XML node identifier.
PATH secondary	Create a secondary XML index on columns built on path values and node values in the primary XML index. In the PATH secondary index, the path and node values are key columns that allow efficient seeks when searching for paths.
VALUE secondary	Create a secondary XML index on columns where key columns are (node value and path) of the primary XML index.
PROPERTY secondary	Create a secondary XML index on columns (PK, path and node value) of the primary XML index where PK is the primary key of the base table.

Primary XML index

Specify the primary XML index to use in creating a secondary XML index.

Advanced Properties for XML Index

Allow row locks

Row locks are allowed when accessing the index. The Database Engine determines when row locks are used.

Allow page locks

Page locks are allowed when accessing the index. The Database Engine determines when page locks are used.

Fill Factor (%)

Specify a percentage that indicates how full the Database Engine should make the leaf level of each index page during index creation or rebuild.

Pad index

The percentage of free space that is specified by fillfactor is applied to the intermediate-level pages of the index.

Sort in tempdb

Specify to store temporary sort results in tempdb.

Max. degree of parallelism

Override the max degree of parallelism configuration option for the duration of the index operation.

General Properties for Spatial Index

Table/View

Must be TABLE.

Table Name

Select a table.

Spatial Column

Select a spatial column which the index is based.

Tessellation Scheme

The tessellation scheme for the spatial index.

Min. Coordinates

Specify the x-coordinate (X) and y-coordinate (Y) of the lower-left corner of the bounding box.

Max. Coordinates

Specify the x-coordinate (X) and y-coordinate (Y) of the upper-right corner of the bounding box.

Level 1

Specify the first (top) level grid.

Level 2

Specify the second-level grid.

Level 3

Specify the third-level grid.

Level 4

Specify the fourth-level grid.

Cells per Object

Specify the number of tessellation cells per object that can be used for a single spatial object in the index by the tessellation process.

Advanced Properties for Spatial Index

Recompute statistics

Enable automatic statistics updating.

Allow row locks

Row locks are allowed when accessing the index. The Database Engine determines when row locks are used.

Note: SQL Azure does not support.

Allow page locks

Page locks are allowed when accessing the index. The Database Engine determines when page locks are used.

Note: SQL Azure does not support.

Fill Factor (%)

Specify a percentage that indicates how full the Database Engine should make the leaf level of each index page during index creation or rebuild.

Note: SQL Azure does not support.

Pad index

The percentage of free space that is specified by fillfactor is applied to the intermediate-level pages of the index.

Note: SQL Azure does not support.

Sort in tempdb

Specify to store temporary sort results in tempdb.

Note: SQL Azure does not support.

Max. degree of parallelism

Override the max degree of parallelism configuration option for the duration of the index operation.

Note: SQL Azure does not support.

SQL Server Synonyms

A synonym is an alternative name for a schema-scoped object. Client applications can use a single-part name to reference a base object by using a synonym instead of using a two-part, three-part, or four-part name to reference the base object. Click -> **Synonym** to open an object list for **Synonym**.

Note: SQL Azure does not support the **Comment** tab.

General Properties

Server Name

The name of the server on which base object is located.

Note: SQL Azure does not support.

Database Name

The name of the database in which the base object is located.

Schema Name

The name of the schema of the base object.

Object Type

The object type.

Object Name

The name of the base object that the synonym references.

SQL Server Triggers

A trigger is a special kind of stored procedure that automatically executes when an event occurs in the database server. Click -> **Trigger** to open an object list for **Trigger**.

See Triggers for details.

Note: SQL Azure does not support the **Comment** tab.

General Properties

Trigger Type

Choose Table or View on which the DML trigger is executed.

Enable

Check this option to enable the trigger.

Table Name or **View name**

Choose a table or a view.

After

Specify that the DML trigger is fired only when all operations specified in the triggering SQL statement have executed successfully.

Instead of

Specify that the DML trigger is executed instead of the triggering SQL statement, therefore, overriding the actions of the triggering statements.

Insert

The trigger is activated whenever a new row is inserted into the table.

Delete

The trigger is activated whenever a row is deleted from the table.

Update

The trigger is activated whenever a row is modified.

SQL statements

Specify additional criteria in Definition tab.

Note: Support from SQL Server 2005 or later.

Assembly method

Specify the method of an assembly to bind with the trigger.

Note: Support from SQL Server 2005 or later.

Advanced Properties

Caller

Specify the statements inside the module are executed in the context of the caller of the module.

Note: Support from SQL Server 2005 or later and SQL Azure.

Owner

Specify the statements inside the module executes in the context of the current owner of the module.

Note: Support from SQL Server 2005 or later and SQL Azure.

Self

Specify the statements inside the module are executed in the context of the specified user is the person creating or altering the module.

Note: Support from SQL Server 2005 or later and SQL Azure.

User

Specify the statements inside the module execute in the context of the user.

Note: Support from SQL Server 2005 or later and SQL Azure.

Encrypted

Obfuscate the text of the CREATE TRIGGER statement.

Note: Support from SQL Server 2005 or later.

Not for replication

Indicate that the trigger should not be executed when a replication agent modifies the table that is involved in the trigger.

Note: SQL Azure does not support.

Append

Specify that an additional trigger of an existing type should be added.

Note: SQL Azure does not support.

Definition

This tab will appear when the **Definition Type** is set to **SQL statements** in General tab or when connecting to SQL Azure. Enter valid SQL statements.

The **Code Outline** window displays information about the trigger. To show the **Code Outline** window, simply choose **Edit** -> **Show Code Outline** from main menu.

Note: Available only in Full Version.

Button	Description
	Refresh the code outline.
	Turn mouse over highlight on or off.
	Show the code outline in a detail way.
	Sort by type and name.
	Expand the selected item.
	Collapse the selected item.

Hint: To customize the view of the editor and find out more features for sql editing, see Editor Advanced Features.

SQL Server Backup Devices

During a backup operation on a SQL Server database, the backed up data (the backup) is written to a physical backup device. This physical backup device is initialized when the first backup in a media set is written to it. The backups on a set of one or more backup devices compose a single media set. Click -> **Backup Device** to open an object list for **Backup Device**.

General Properties

Destination

Specify the physical file name or path of the backup device.

Type

The type of the backup device: DISK.

SQL Server Linked Servers

A linked server configuration enables SQL Server to execute commands against OLE DB data sources on remote servers. Click -> **Linked Server** to open an object list for **Linked Server**.

Linked servers offer the following advantages:

- Remote server access.
- The ability to issue distributed queries, updates, commands, and transactions on heterogeneous data sources across the enterprise.
- The ability to address diverse data sources similarly.

Note: SQL Azure does not support.

General Properties

You can choose the **Server Type**: **SQL Server** or **Other data source**. If you choose **Other data source**, define the required information:

Provider

Choose the unique programmatic identifier (PROGID) of the OLE DB provider corresponding to the data source.

Product Name

Define the product name of the OLE DB data source to add as a linked server.

Data Source

Define the name of the data source as interpreted by the OLE DB provider.

Provider String

Define the OLE DB provider-specific connection string that identifies a unique data source.

Location

Define the location of the database as interpreted by the OLE DB provider.

Catalog

Define the catalog to be used when making a connection to the OLE DB provider.

Security

In this tab, add or delete a mapping between logins on the local instance of SQL Server and remote logins on the linked server.

Local Login

Choose a login on the local server.

Impersonate

Check this option to specify that logins use their own credentials to connect to the linked server.

Remote User

Enter the username used to connect the linked server.

Remote Password

Enter the user password.

Set the action when a login not defined in the list:

- not be made
- be made without using a security context
- be made using the login's current security context
- be made using the following security context

  Set the **Remote login** and **Password**

Advanced Properties

Connect timeout

Define the time-out value for connecting to a linked server. If 0, use the sp_configure default.

Query timeout

Define the time-out value for queries against a linked server. If 0, use the sp_configure default.

Data Access

Check this option to enable a linked server for distributed query access.

Collation Compatible

If this option is checked, SQL Server assumes that all characters in the linked server are compatible with the local server, with regard to character set and collation sequence (or sort order). This enables SQL Server to send comparisons on character columns to the provider.

Use Remote Collation

If this option is checked, the collation of remote columns is used for SQL Server data sources, and the collation specified in collation name is used for non-SQL Server data sources.

Collation

Specify the name of the collation used by the remote data source if Use Remote Collation is checked and the data source is not a SQL Server data source. The name must be one of the collations supported by SQL Server.

Lazy Schema Validation

If this option is checked, skip schema checking of remote tables at the beginning of the query.

Publisher

Check this option to enable publisher.

Subscriber

Check this option to enable subscriber.

Distributor

Check this option to enable distributor.

RPC

Check this option to enable RPC from the given server.

RPC Out

Check this option to enable RPC to the given server.

Promotion of Distributed Transactions for RPC

Use this option to protect the actions of a server-to-server procedure through a Microsoft Distributed Transaction Coordinator (MS DTC) transaction.

Note: Support from SQL Server 2005 or later.

SQL Server Server Triggers

A server trigger can be a DDL or logon trigger for current server. DDL triggers execute in response to a variety of data definition language (DDL) events. These events primarily correspond to Transact-SQL CREATE, ALTER, and DROP statements, and certain system stored procedures that perform DDL-like operations. Logon triggers fire in response to the LOGON event that is raised when a user sessions is being established. Click -> **Server Trigger** to open an object list for **Server Trigger**.

Note: Support from SQL Server 2005 or later.

General Properties

Trigger Type

Choose the trigger type.

Enable

Check this option to enable the trigger.

SQL statements

Specify additional criteria in Definition tab.

Assembly method

Specify the method of an assembly to bind with the trigger.

Event

Check the DDL event from the list.

Advanced Properties

Caller

Specify the statements inside the module are executed in the context of the caller of the module.

Self

Specify the statements inside the module are executed in the context of the specified user is the person creating or altering the module.

Login

Specify the statements inside the module execute in the context of the user.

Encrypted

Obfuscate the text of the CREATE TRIGGER statement.

Definition

This tab will appear when the **Definition Type** is set to **SQL statements** in General tab or when connecting to SQL Azure. Enter valid SQL statements.

The **Code Outline** window displays information about the trigger. To show the **Code Outline** window, simply choose **Edit** -> **Show Code Outline** from main menu.

Note: Available only in Full Version.

Button	Description
	Refresh the code outline.
	Turn mouse over highlight on or off.
	Show the code outline in a detail way.
	Sort by type and name.
	Expand the selected item.
	Collapse the selected item.

Hint: To customize the view of the editor and find out more features for sql editing, see Editor Advanced Features.

SQL Server Assemblies

An assembly is a managed application module that contains class metadata and managed code as an object in an instance of SQL Server. By referencing this module, common language runtime (CLR) functions, stored procedures, triggers, user-defined aggregates, and user-defined types can be created in the database. Click -> **Assembly** to open an object list for **Assembly**.

Note: Support from SQL Server 2005 or later.

General Properties

Owner

Specify the name of a user or role as owner of the assembly.

Unchecked data

This option allows postponing the checks until a later time by using DBCC CHECKTABLE.

Permission Set

Specify a set of code access permissions that are granted to the assembly when it is accessed by SQL Server. If not specified, SAFE is applied as the default.

Visible

Indicate whether the assembly is visible for creating common language runtime (CLR) functions, stored procedures, triggers, user-defined types, and user-defined aggregate functions against it.

Assembly

Specify the local path or network location where the assembly that is being uploaded is located, and also the manifest file name that corresponds to the assembly.

Dependent Assemblies

Upload a file to be associated with the assembly, such as source code, debug files or other related information, into the server and made visible in the sys.assembly_files catalog view.

SQL Server Database Triggers

A database trigger is a DDL trigger to the current database. DDL triggers execute in response to a variety of data definition language (DDL) events. These events primarily correspond to Transact-SQL CREATE, ALTER, and DROP statements, and certain system stored procedures that perform DDL-like operations. Click -> **Database Trigger** to open an object list for **Database Trigger**.

Note: Support from SQL Server 2005 or later and SQL Azure.

SQL Azure does not support the **Comment** tab.

General Properties

Trigger Type

Trigger type must be Database Trigger.

Enable

Check this option to enable the trigger.

SQL statements

Specify additional criteria in Definition tab.

Note: SQL Azure does not support.

Assembly method

Specify the method of an assembly to bind with the trigger.

Note: SQL Azure does not support.

Event

Check the DDL event from the list.

Advanced Properties

Caller

Specify the statements inside the module are executed in the context of the caller of the module.

Self

Specify the statements inside the module are executed in the context of the specified user is the person creating or altering the module.

User

Specify the statements inside the module execute in the context of the user.

Encrypted

Obfuscate the text of the CREATE TRIGGER statement.

Note: SQL Azure does not support.

Definition

This tab will appear when the **Definition Type** is set to **SQL statements** in General tab or when connecting to SQL Azure. Enter valid SQL statements.

The **Code Outline** window displays information about the trigger. To show the **Code Outline** window, simply choose **Edit** -> **Show Code Outline** from main menu.

Note: Available only in Full Version.

Button	Description
	Refresh the code outline.
	Turn mouse over highlight on or off.
	Show the code outline in a detail way.
	Sort by type and name.
	Expand the selected item.
	Collapse the selected item.

Hint: To customize the view of the editor and find out more features for sql editing, see Editor Advanced Features.

SQL Server Partition Functions

A partition function is a function in the current database that maps the rows of a table or index into partitions based on the values of a specified column. Click -> **Partition Function** to open an object list for **Partition Function**.

Note: Support from SQL Server 2005 or later.

General Properties

Type

Choose the data type of the column used for partitioning.

Size

Specify the size of the data type if necessary.

Scale

Specify the scale of the data type if necessary.

Collation

Specify the collation of the data type if necessary.

Boundary values belong to right interval

Specify to the right of each boundary value interval.

Boundary values

Specify the boundary values for each partition of a partitioned table or index that uses partition_function_name.

SQL Server Partition Schemes

A partition scheme is a scheme in the current database that maps the partitions of a partitioned table or index to filegroups. The number and domain of the partitions of a partitioned table or index are determined in a partition scheme. Click -> **Partition Scheme** to open an object list for **Partition Scheme**.

Note: Support from SQL Server 2005 or later.

General Properties

Partition Function

Choose the partition function.

Filegroups

Specify the filegroups to hold the partitions specified by partition_function_name.

SQL Preview

The **SQL Preview** tab shows the CREATE statement and necessary SQL statements of the database or schema object.

For some database or schema objects, you can use the below dropdown list to show the SQL which will be run when pressing **Save** or **Save As** button.

Maintain

Navicat provides a complete solution for maintaining databases and their database objects in MySQL, Oracle, PostgreSQL, SQLite, SQL Server and MariaDB.

To maintain server objects, you can control-click it and select **Maintain** from the popup menu.

Maintain MySQL/MariaDB

Maintain Table

Analyze Table

Analyze and store the key distribution for the table. During the analysis, the table is locked with a read lock for MyISAM and BDB. For InnoDB the table is locked with a write lock. Currently, MySQL supports analyzing only for MyISAM, BDB, and InnoDB tables.

Check Table

Check a table or tables for errors. Currently, MySQL supports checking only for MyISAM, InnoDB and ARCHIVE tables. For MyISAM tables, the key statistics are updated as well.

Normal	Run the CHECK TABLE statement without an extra option.
Quick	Don't scan the rows to check for wrong links.
Fast	Don't scan the rows to check for wrong links.
Changed	Only check tables which have been changed since last check or haven't been closed properly.
Extended	Do a full key lookup for all keys for each row. This ensures that the table is 100 % consistent, but will take a long time.

Optimize Table

The main reason for optimizing your table is to reclaim unused space and to defragment the data file. You should optimize a table if you have deleted a large part of a table or if you have made many changes to a table with variable-length rows (tables that have VARCHAR, BLOB, or TEXT columns). Deleted records are maintained in a linked list and subsequent INSERT operations reuse old row positions. Currently, MySQL supports optimizing only for MyISAM, InnoDB and BDB tables.

Repair Table

Repair a possibly corrupted table and returns a result set.

Quick	Repair Table tries to repair only the index tree.
Extended	MySQL creates the index row by row instead of creating one index at a time with sorting.

Maintain Oracle

Maintain Table

Enable Table Lock

Choose Enable Table Lock to enable table locks, thereby allowing DDL operations on the table. All currently executing transactions must commit or roll back before Oracle Database enables the table lock.

Disable Table Lock

Choose Disable Table Lock to disable table locks, thereby preventing DDL operations on the table.

Enable Row Movement

Choose Enable Row Movement to allow the database to move a row, thus changing the rowid.

Disable Row Movement

Choose Disable Row Movement if you want to prevent the database from moving a row, thus preventing a change of rowid.

Shrink Space

Shrink Space is to compact the table segment. This clause is valid only for segments in tablespaces with automatic segment management. By default, Oracle database compacts the segment, adjusts the high water mark, and releases the recuperated space immediately.

Compacting the segment requires row movement. Therefore, you must enable row movement for the table you want to shrink before shrink space. Further, if your application has any rowid-based triggers, you should disable them before issuing this clause.

Move

Move relocates data of a nonpartitioned table or of a partition of a partitioned table into a new segment, optionally in a different tablespace, and optionally modify any of its storage attributes.

Validate Table Structure

Validate Structure verifies the integrity of the structure of a table. The statistics collected by this clause are not used by the Oracle database optimizer. If the structure is valid, no error is returned. However, if the structure is corrupt, an error message will be shown.

For a table, Oracle database verifies the integrity of each of the data blocks and rows.

Collect Table Statistics

Collect Statistics analyzes the contents of tables. When you analyze a table, the database collects statistics about expressions occurring in any function-based indexes as well. Therefore, be sure to create function-based indexes on the table before analyzing the table.

Maintain View

Compile

To recompile the view specification or body.

Maintain Function/Procedure

Compile

To recompile the function/procedure specification or body.

Compile for Debug

To recompile the function/procedure specification or body and instruct the PL/SQL compiler to generate and store the code for use by the PL/SQL debugger.

Maintain Index

Rebuild

To re-create an existing index or one of its partitions or subpartitions. If the index is marked unusable, then a successful rebuild will mark it usable.

Make Index Unusable

To make the index unusable. An unusable index must be rebuilt, or dropped and re-created, before it can be used.

Coalesce Index

To instruct Oracle database to merge the contents of index blocks where possible to free blocks for reuse.

Compute Index Statistics

To compute the statistics of the index.

Maintain Java

Compile or Resolve

To resolve the primary Java class schema object.

Set AuthID Current User

Set the invoker rights to AUTHID CURRENT_USER.

Set AuthID Definer

Set the invoker rights to AUTHID DEFINER.

Maintain Materialized View

Enable Row Movement

To enable row movement.

Disable Row Movement

To disable row movement.

Shrink Space

To compact the materialized view segment. By default, Oracle database compacts the segment, adjusts the high water mark, and releases the recuperated space immediately.

Compile

To explicitly revalidate a materialized view. If an object upon which the materialized view depends is dropped or altered, then the materialized view remains accessible, but it is invalid for query rewrite. You can choose this option to explicitly revalidate the materialized view to make it eligible for query rewrite.

Force Materialized View Refresh

To perform a refresh.

Maintain Materialized View Log

Enable Row Movement

To enable row movement. Row movement indicates that rowids will change after the flashback occurs.

Disable Row Movement

To disable row movement.

Shrink Space

To compact the materialized view log segments. By default, Oracle database compacts the segment, adjusts the high water mark, and releases the recuperated space immediately.

Maintain Package

Compile

To recompile the package specification or body.

Compile for Debug

To recompile the package specification or body and instruct the PL/SQL compiler to generate and store the code for use by the PL/SQL debugger.

Maintain Trigger

Compile

To explicitly compile the trigger, whether it is valid or invalid. Explicit recompilation eliminates the need for implicit run-time recompilation and prevents associated run-time compilation errors and performance overhead.

Compile for Debug

To recompile the trigger and instruct the PL/SQL compiler to generate and store the code for use by the PL/SQL debugger.

Enable Trigger

To enable the trigger.

Disable Trigger

To disable the trigger.

Maintain Type

Compile

To compile the type specification and body.

Compile for Debug

To recompile the type specification or body and instruct the PL/SQL compiler to generate and store the code for use by the PL/SQL debugger.

Maintain XML Schema

Compile

To re-compile an already registered XML schema. This is useful for bringing a schema in an invalid state to a valid state.

Purge XML Schema

To remove the XML Schema completely from Oracle XML DB in Oracle 11g.

Maintain Tablespace

Read Only

To place the tablespace in transition read-only mode. In this state, existing transactions can complete (commit or roll back), but no further DML operations are allowed to the tablespace except for rollback of existing transactions that previously modified blocks in the tablespace.

Read Write

To indicate that write operations are allowed on a previously read-only tablespace.

Online

To take the tablespace online.

Offline

To take the tablespace offline.

Normal	To flush all blocks in all datafiles in the tablespace out of the system global area (SGA).
Temporary	Oracle database performs a checkpoint for all online datafiles in the tablespace but does not ensure that all files can be written.
Immediate	Oracle database does not ensure that tablespace files are available and does not perform a checkpoint.

Coalesce

To combine all contiguous free extents into larger contiguous extents for each datafile in the tablespace.

Shrink Space

To reduce the amount of space the tablespace is taking. This is valid only for temporary tablespaces in Oracle 11g.

Maintain User

Lock User

To lock user account.

Unlock User

To unlock user account.

Expire User

To set user account will expire.

Maintain PostgreSQL

Maintain Database and Table

Analyze Database and **Analyze Table**

Collect statistics about the contents of tables in the database, and stores the results in the system table *pg_statistic*. Subsequently, the query planner uses these statistics to help determine the most efficient execution plans for queries. **Analyze Database** examines every table in the current database.

When VERBOSE is specified, ANALYZE emits progress messages to indicate which table is currently being processed. Various statistics about the tables are printed as well. It is enabled in Navicat by default.

Vacuum Database and **Vacuum Table**

Reclaim storage occupied by deleted tuples. In normal PostgreSQL operation, tuples that are deleted or obsoleted by an update are not physically removed from their table; they remain present until a Vacuum is done. Therefore it's necessary to do Vacuum periodically, especially on frequently-updated tables. **Vacuum Database** examines every table in the current database.

When VERBOSE is specified, VACUUM emits progress messages to indicate which table is currently being processed. Various statistics about the tables are printed as well. It is enabled in Navicat by default.

Plain	Run the VACUUM VERBOSE statement without extra options.
Full	Select "full" vacuum, which may reclaim more space, but takes much longer and exclusively locks the table.
Freeze	Select aggressive "freezing" of tuples.
Plain Analyze	Update statistics used by the planner to determine the most efficient way to execute a query.
Full Analyze	Select "full" vacuum, which may reclaim more space, but takes much longer and exclusively locks the table. Update statistics used by the planner to determine the most efficient way to execute a query.
Freeze Analyze	Select aggressive "freezing" of tuples. Update statistics used by the planner to determine the most efficient way to execute a query.

Reindex Database and **Reindex Table**

Rebuild an index using the data stored in the index's table, replacing the old copy of the index. There are several scenarios in which to use Reindex:

- An index has become corrupted, and no longer contains valid data.
- An index has become "bloated", that it is contains many empty or nearly-empty pages.
- You have altered a storage parameter (such as fill factor) for an index, and wish to ensure that the change has taken full effect.
- An index build with the CONCURRENTLY option failed, leaving an "invalid" index.

Maintain SQLite

Maintain Database and Table

Analyze Database and **Analyze Table**

Gather statistics about tables and indexes and stores the collected information in internal tables of the database where the query optimizer can access the information and use it to help make better query planning choices.

Vacuum Database

Rebuild the entire database. VACUUM only works on the main database. It is not possible to VACUUM an attached database file.

Reindex Database and **Reindex Table**

Delete and recreate all indexes in database or attached to the tables from scratch. This is useful when the definition of a collation sequence has changed.

Maintain Index

Reindex

Delete and recreate the index from scratch. This is useful when the definition of a collation sequence has changed.

Maintain SQL Server

Maintain Database

Read-Write

When READ_WRITE is specified, users can retrieve and modify data. READ_WRITE is the default setting.

Read-Only

When READ_ONLY is specified, the database is in read-only mode.

Online

When ONLINE is specified, the database is open and available for use. ONLINE is the default setting.

Offine

When OFFLINE is specified, the database is closed and shutdown cleanly and marked offline. The database cannot be modified while the database is offline.

Emergency

User has changed the database and set the status to EMERGENCY. The database is in single-user mode and may be repaired or restored. The database is marked READ_ONLY, logging is disabled, and access is limited to members of the sysadmin fixed server role. EMERGENCY is primarily used for troubleshooting purposes. For example, a database marked as suspect can be set to the EMERGENCY state. This could permit the system administrator read-only access to the database. Only members of the sysadmin fixed server role can set a database to the EMERGENCY state.

Multi User

MULTI_USER allows all users with the appropriate permissions to connect to the database. MULTI_USER is the default setting.

Single User

SINGLE_USER allows one user at a time to connect to the database. All other user connections are broken. The timeframe for breaking the connection is controlled by the termination clause of the ALTER DATABASE statement. New connection attempts are refused. The database remains in SINGLE_USER mode even if the user who set the option logs off. At that point, a different user (but only one) can connect to the database.

Restricted User

RESTRICTED_USER allows only members of the db_owner fixed database role and dbcreator and sysadmin fixed server roles to connect to the database, but it does not limit their number. Users who are not members of these roles are disconnected in the timeframe specified by the termination clause of the ALTER DATABASE statement. Moreover, new connection attempts by unqualified users are refused.

Maintain Assembly

Set Visible

To show the assembly.

Set Invisible

To hide the assembly.

Maintain Index

Rebuild Index

To rebuild and enable the index.

Reorganize Index

To reorganize the enabled index.

Disable Index

To disable the index.

Maintain Trigger

Enable Trigger

To enable the server trigger, database trigger or trigger.

Disable Trigger

To disable the server trigger, database trigger or trigger.

Maintain Login

Enable Login

To enable the login.

Disable Login

To disable the login.

Table Viewer

Table Viewer displays the table data as a grid. Data can be displayed in two modes: **Grid View** and **Form View**.

The toolbars of Table Viewer provides the following functions for managing data:

- **Begin Transaction/Commit/Rollback**

  Click **Begin Transaction** to start a transaction. To make permanent all changes performed in the transaction, click **Commit**. Or, click **Rollback** to undo work done in the current transaction.

  Hint: The **Commit** and **Rollback** buttons are available only when **Auto begin transaction** is enabled under Preferences or after clicking the **Begin Transaction** button.

- Edit TEXT/BLOB/BFile

  Allow you to view and edit the content of TEXT, BLOB and BFile fields.

  Note: Only Oracle supports BFile.

- Filter Data

  Allow you to filter records by creating and applying filter criteria for the data grid.

- Import Data

  Import data from files.

- Export Data

  Export data to files.

- Sort Records

  Sort Records by custom order.

Grid View

The Grid View allows you to view, update, insert, or delete data in a table. The popup menu of the grid provides the following additional functions: set the field value as Null, use current field value as a filter, format grid view, and more.

Using Navigation Bar

Table Viewer provides a convenient way to navigate among the records/pages using **Record/Page Navigation Bar** buttons. All buttons are used to navigate left and right to the previous or the next records/pages.

Record Navigation Bar

Button	Description
+	New record: enter a new record. At any point when you are working with your table in the grid view, click on this button to get a blank display for a record.
−	Delete record: delete an existing record.
✔	Apply changes: apply the changes.
✖	Cancel changes: remove all edits made to the current record.
↻	Refresh: refresh the table.

■	Stop: stop when loading enormous data from server.

Note: The SQL statement shows next to the Page Navigation Bar indicates any statement has just been executed.

Page Navigation Bar

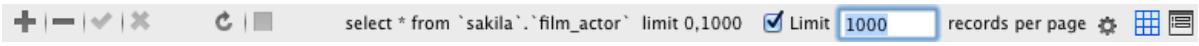
Button	Description
⇤	First Page: move to first page.
←	Previous Page: move to previous page.
→	Next Page: move to next page.
⇥	Last Page: move to last page.
⚙	Limit Record Setting: set number of records showing on each page.
▦	Grid View: switch to grid view.
▤	Form View: switch to form view.

Use the **Limit Record Setting** ⚙ button to enter to the edit mode.

Limit ☐ records per page

Check this option if you want to limit the number of records showed on each page. Otherwise, all records will be displayed in one single page. And, set the **records per page** value in the edit field. The number representing the number of records showed per page.

Note: This setting mode will take effect on current table only. To adjust the global settings, see Preferences.



Extra Navigation Bar for Form View

Button	Description
⇤	First Record: move to the first record.
←	Previous Record: move one record back (if there is one) from the current record.
→	Next Record: move one record ahead.
⇥	Last Record: move to the last record.

Editing Records

The navigation bar allows you to switch the records quickly, insert, update or delete records. View data as a grid is most helpful for entering new records and editing old records in a table.

To add a record

1. Make sure that your cursor is situated in the first blank cell on the table, then enter the desired data. If you are adding the new record into an existing table, just simply press CMD-+ to get a blank display for a record.
2. Watch the graphics symbol in the record selectors box just to the left of your record. It will change from the , which indicates that it is the current record, to , which indicates that you are editing this record.
3. Just simply move to another record to save the record.

To edit a record

1. Select the record that you wish to edit by clicking in the specific field you want to change.
2. Type in the new data for that field.
3. Just simply move to another record, the new data will overwrite the previous data.

Note: Close the table is another way to save the records.

To delete a record

1. Select the record that you wish to delete.
2. Just simply control-click and select **Delete Row**.

Edit Records with Special Handling

To edit the text field record, just simply click **Show Text in Grid** from the **Edit** menu.

RecordID	LAST_NAME	FIRST_NAME
1	[TEXT]	Jennifer
2	[TEXT]	Arthur
3	[TEXT]	Debra
4	[TEXT]	Dave
5	[TEXT]	Cindy

Hint: To view/edit the text field record in an ease way, see Memo Editor.

To view images in the grid, just simply click **Show Image in Grid** from the **Edit** menu.

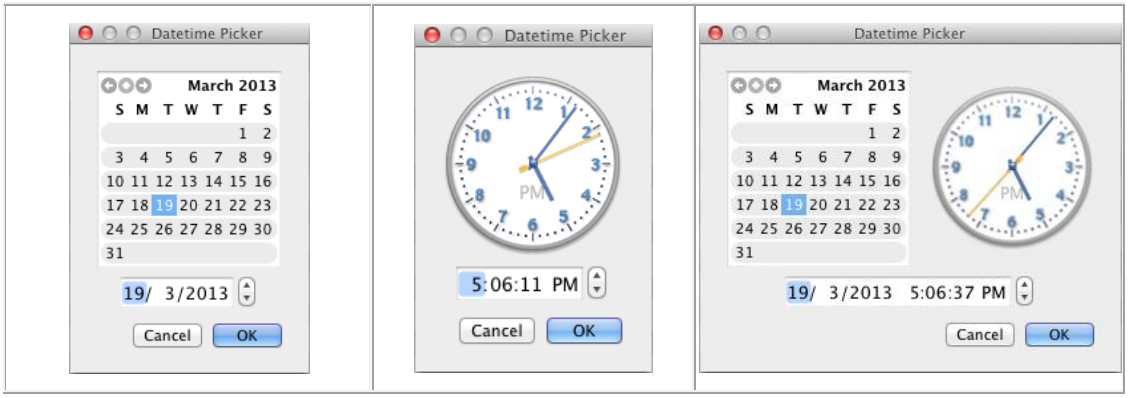
id	image
1	
2	
3	

Hint: To view/edit the image in an ease way, see Image Editor.

To edit a Date/Time record, just simply click to open the editor for editing. Choose/enter the desired data. The editor used in cell is determined by the field type assigned to the column.

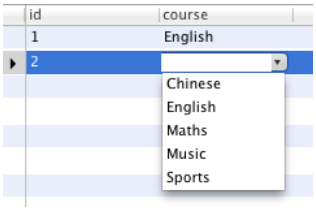
Note: Available only for MySQL, Oracle, PostgreSQL, SQL Server and MariaDB.

Date	Time	DateTime/Timestamp



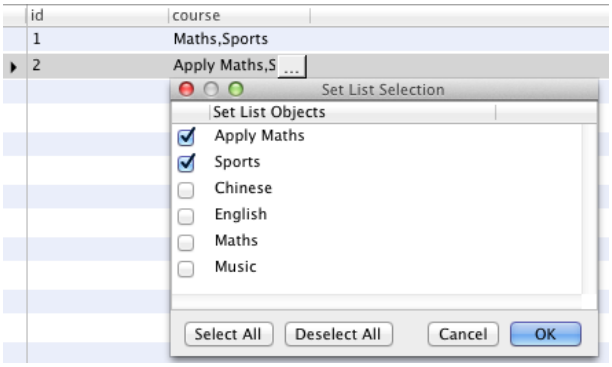
To edit a Enum record, just simply choose the record from the drop-down list.

Note: Available only for MySQL and MariaDB.



To edit a Set record, just simply click ... to open the editor for editing. Select the record(s) from the list. To remove the records, uncheck them in the same way.

Note: Available only for MySQL and MariaDB.



To view BFile content, just simply enable **Preview BFILE** under the **Edit** menu.

Note: Available only for Oracle.

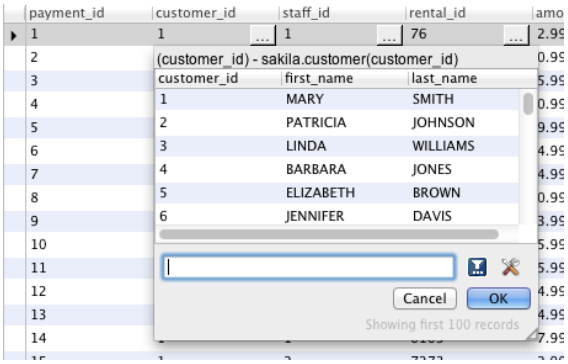
To generate UUID/GUID, control-click the selected cell and select **Generate UUID**.

Note: Available only for PostgreSQL and SQL Server.

Edit Records with Foreign Key (Foreign Key Data Selection - Available only in Full Version)

Foreign Key Data Selection is a useful tool for letting you to get the available value from the reference table in an easy way. It allows you to show additional record(s) from the reference table and search for a particular record(s).

To include data to the record, just simply click ... to open the editor for editing.

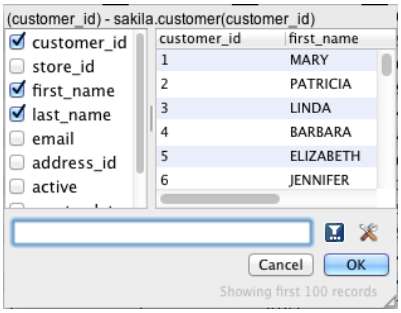


Just simply double-click to select the desired data.

Hint: By default, the number of records showed per page is **100**. To show all records, control-click anywhere on the grid and select **Show All**. To adjust the global settings, see Preference.

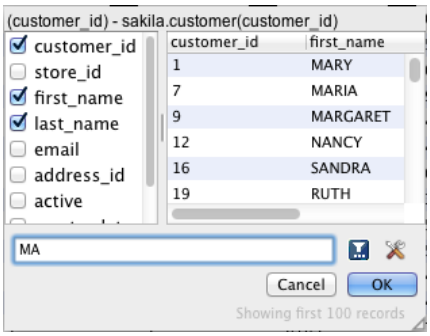
To refresh the record, control-click anywhere on the grid and select **Refresh**.

Click to open a panel on the left for showing a list of column name(s). Just simply click to show the additional column. To remove the column(s), uncheck them in the same way.



Hint: To find for the text in the editor window, control-click anywhere on the grid and select **Find** or press CMD-F.

Enter a value into the edit box and click to filter for the particular record(s).



Hint: To remove the filter results, control-click anywhere on the grid and select **Show All**.

Copy Data from Navicat

Data that being copied from Navicat goes into the windows clipboard with the fields delimited by tabs and the records delimited by carriage returns. It allows you to easily paste the clipboard contents into any application you want. Spreadsheet applications in general will notice the tab character between the fields and will neatly separate the clipboard data into rows and columns.

To select data using **Keyboard Shortcuts**

CMD-A	Toggle the selection of all rows and columns in a data grid.
SHIFT-UP ARROW	Toggle the selection of rows as you move up in the data grid.
SHIFT-DOWN ARROW	Toggle the selection of rows in the data grid as you move down.

To select data using **Mouse Actions**

- Highlighted the desired records by holding down the CMD key while clicking on each row.
- Highlighted range of records by clicking the first row you want to select and holding down the SHIFT key together with moving your cursor to the last row you wish to select.

Note: After you have selected the desired records, just simply press CMD-C or select the **Copy** under the **Edit** menu.

Paste Data into Navicat

Data is copied into the clipboard will be arranged as below format:

1. Data is arranged into rows and column.
2. Rows and columns are delimited by carriage returns/tab respectively.
3. Columns in the clipboard have the same sequence as the columns in the data grid you have selected.

When pasting data into Navicat, you can replace the contents of current records and append the clipboard data into the table. To replace the contents of current records in a table, one must select the rows in the data grid whose contents must be replaced by the data in the clipboard.

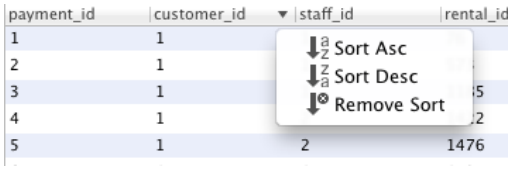
Note: Just simply press CMD-V or select the **Paste** under the **Edit** menu. The paste action cannot be undone if you do not enable transaction.

Sorting/Finding/Replacing Records

Sorting Records

Server stores records in the order they were added to the table. Sorting in Navicat is used to temporarily rearrange records, so that you can view or update them in a different sequence.

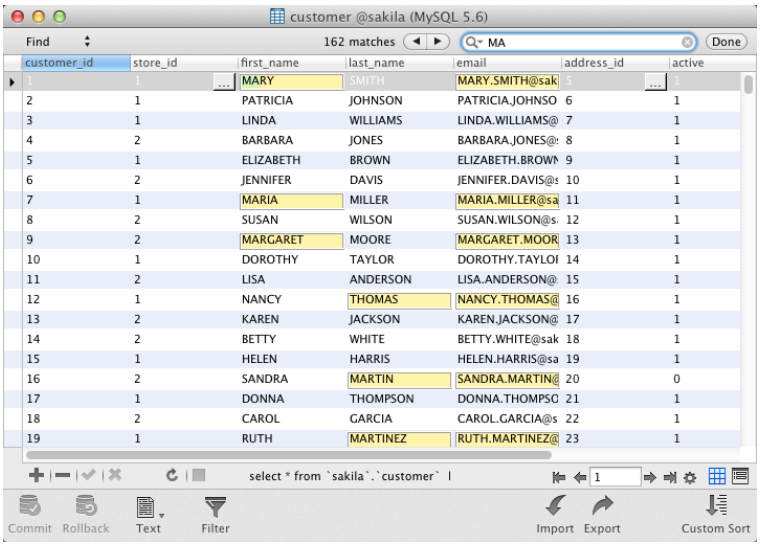
Move over the column caption whose contents you want to sort by, click the right side of the column and select the **Sort Asc**, **Sort Des** or **Remove Sort**.



To sort by custom order of multi fields, click the **Custom Sort** from the toolbar. Then, set the sorting fields and the sorting order.

Finding Records

The **Find** bar is provided for quick searching for the text in the editor window. Just simply click **Edit** -> **Find** -> **Find** from the menu or press CMD-F. Then, click and choose **Find Data** and enter a search string.



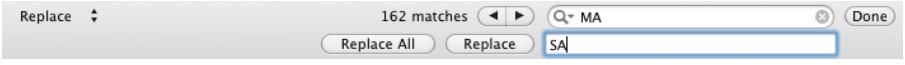
The search starts at the cursor's current position to the end of the file. There will not have differentiates when performing a uppercase or lowercase search.

To find for the next text, just simply select **Edit** -> **Find** -> **Find Next** from the menu or press CMD-G.

Replacing Records

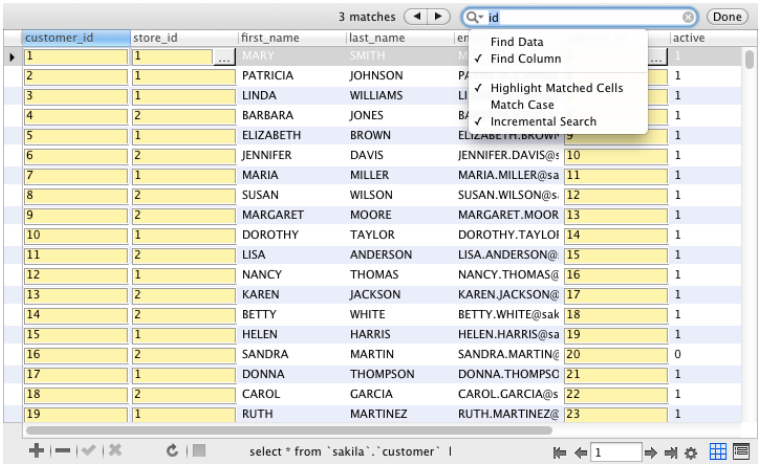
To open the **Replace** option, just simply click **Edit** -> **Find** -> **Find and Replace** from the menu or press OPTION-CMD-F.

Click **Replace** or **Replace All** button to replace the first occurrence or all occurrences automatically.



Finding Columns

To search a column, just simply click **Edit** -> **Find** -> **Find** from the menu or press CMD-F. Then, click and choose **Find Column** and enter a search string.



Filtering Records (Available only in Full Version)

Use either of the following methods to filter the data in the grid:

- The **Custom Filter** Dialog is provided for quick building a simple filter. Just simply control-click a field and select the **Filter** -> **Custom Filter** from the popup menu. Use character '**_**' to represent any single symbol in the condition and use character '**%**' to represent any series of symbols in the condition.

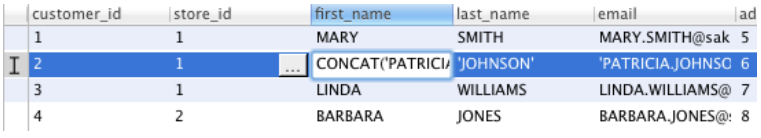
- You can also customize your filter in a more complicated way by control-click a field and selecting the **Filter** -> **Filter Wizard** from the popup menu or clicking the **Filter** from the toolbar. The Filter Wizard becomes visible at the top of grid, where you can see the active filtering condition and easily enable or disable it by clicking a check box at the left.

Manipulating Raw Data

Navicat normally recognize what user has input in grid as normal string, any special characters or functions would be processed as plain text (that is, its functionality would be skipped).

Editing data in **Raw Mode** provides an ease and direct method to apply server built-in function. To access the Raw Mode function, control-click a field and select **Raw** from the pop-up menu.

Note: Available only for MySQL, PostgreSQL, SQLite, SQL Server and MariaDB.



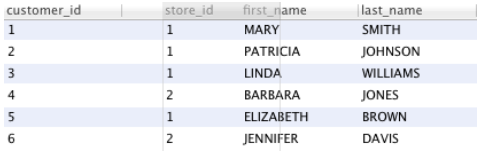
Formatting Table Grid

Use the following methods to format the table grid:

Move Columns

1. Click on the column header and hold down the left mouse button.

2. Move the pointer to the desired location.
3. Release the mouse and the column will move.



customer_id	store_id	first_name	last_name
1	1	MARY	SMITH
2	1	PATRICIA	JOHNSON
3	1	LINDA	WILLIAMS
4	2	BARBARA	JONES
5	1	ELIZABETH	BROWN
6	2	JENNIFER	DAVIS

Set Column Width

Click right border at top of column and drag either left or right.

Control-click the column you want to set the column width with and select **Set Column Width** or select **Edit** -> **Set Column Width** from the menu. Specify width in the **Set Column Width** Dialog. The default value is 100.

Hint: To adjust the width for all columns, select **Edit** -> **Set All Columns Width** from the menu. To adjust the global settings, see Preferences.

Set Row Height

Control-click anywhere on the table grid and select **Set Row Height** or select **Edit** -> **Set Row Height** from the menu. Specify row height in the **Set Row Height** Dialog. The default value is 17.

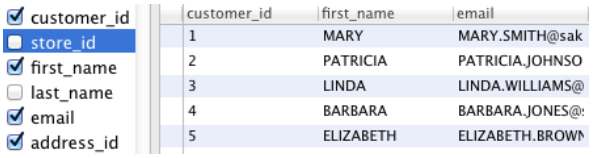
Hint: This action applies on the current table grid only. To adjust the global settings, see Preferences.

Show/Hide Columns

If there are many columns in the table and you want to hide some of them from the table grid. Just simply control-click anywhere on the table grid and select **Show/Hide Columns** or select **Edit** -> **Show/Hide Columns** from the menu. Select the columns that you would like to hide.

The hidden column(s) will disappear from the table grid.

To unhide the columns, select **Edit** -> **Show/Hide Columns** from the menu. Select the columns that you would like to redisplay.



Show/Hide ROWID

If you want to display or hide the rowid (address) of every row, select **Edit** -> **Show/Hide ROWID** from the menu.

The column **ROWID** will be showed in the last column.

Note: Available only for Oracle and SQLite.

Form View (Available only in Full Version)

The Form View allows you to view, update, insert, or delete data as a form, which the current record is displayed: field name and its value. The popup menu of the form provides the following additional functions: set the field value as Null/Empty String, use current field value as a filter, format form view, and more.

The navigation bar allows you to switch the records quickly, insert, update or delete records.

Related topic:

Sorting/Finding/Replacing Records

Filtering Records

Manipulating Raw Data

Formatting Table Grid

Assistant Editors

Navicat provides Text/Hex/Image pane to view and edit TEXT/BLOB/BFile fields content. The editor allows you to view, update, insert, or delete data in a table. Click **Text**, **Hex** and **Image** from the toolbar to activate the appropriate viewer/editor.

Note: Oracle BFile fields cannot be edited.

The **Text** pane allows you to edit data as a simple text. Use the button on the navigation bar to update the changed records to the table.

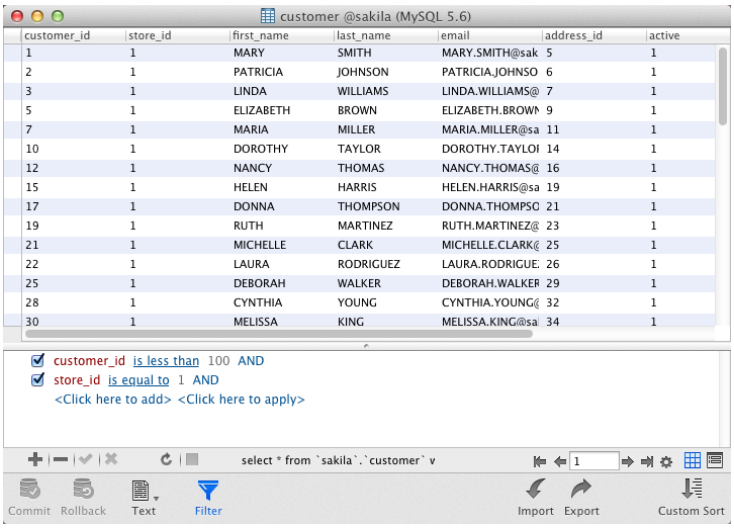
The **Hex** pane allows you to edit data in hexadecimal mode. Use the button on the navigation bar to update the changed records to the table.

The **Image** pane allows you to show data as image. Use the **Load**, **Save to disk** and **Clear** button to load/remove the image from a file, or save the image to a table.

Note: You can simply drag and drop an image file to the image editor.

Filter Wizard (Available only in Full Version)

Filter Wizard allows you to facilitate creating and applying filter criteria that you specify for the table grid. Moreover, it allows you to save filter criteria as a profile for future use. Click **Filter** from the toolbar to activate the editor.


customer @sakila (MySQL 5.6)
customer_id store_id first_name last_name email address_id active
1 1 MARY SMITH MARY.SMITH@sak 5 1
2 1 PATRICIA JOHNSON PATRICIA.JOHNSO 6 1
3 1 LINDA WILLIAMS LINDA.WILLIAMS@ 7 1
5 1 ELIZABETH BROWN ELIZABETH.BROWN 9 1
7 1 MARIA MILLER MARIA.MILLER@sa 11 1
10 1 DOROTHY TAYLOR DOROTHY.TAYLOI 14 1
12 1 NANCY THOMAS NANCY.THOMAS@ 16 1
15 1 HELEN HARRIS HELEN.HARRIS@sa 19 1
17 1 DONNA THOMPSON DONNA.THOMPSO 21 1
19 1 RUTH MARTINEZ RUTH.MARTINEZ@ 23 1
21 1 MICHELLE CLARK MICHELLE.CLARK@ 25 1
22 1 LAURA RODRIGUEZ LAURA.RODRIGUE 26 1
25 1 DEBORAH WALKER DEBORAH.WALKER 29 1
28 1 CYNTHIA YOUNG CYNTHIA.YOUNG@ 32 1
30 1 MELISSA KING MELISSA.KING@sa 34 1
customer_id is less than 100 AND
store_id is equal to 1 AND
<Click here to add> <Click here to apply>
select * from `sakila`.`customer` v
Commit Rollback Text Filter Import Export Custom Sort

1. To add a new condition to the criteria, just simply click the **<Click here to add>**.
2. To choose the column, click on the column box (next to the check box) and select an item from the dropdown list.
3. Click on the filter operator box (next to the column box) and select an item from the dropdown list. You can choose **Custom** from the list to enter the condition manually.

Filter Operator	Result
begins with <?>	My_Field LIKE 'your_value%'
contains <?>	My_Field LIKE '%your_value%'
does not contain <?>	NOT (My_Field LIKE '%your_value%')
ends with <?>	My_Field LIKE '%your_value'
is between <?> <?>	((My_Field >= your_value1) and (My_Field <= your_value2))
is blank	My_Field = ''
is equal to <?>	My_Field = 'your_value'
is greater than <?>	My_Field > 'your_value'
is greater than or equal to <?>	My_Field >= 'your_value'
is less than <?>	My_Field < 'your_value'
is less than or equal to <?>	My_Field <= 'your_value'
is not between <?> <?>	NOT ((My_Field >= your_value1) and (My_Field <= your_value2))
is not blank	My_Field <> ''
is not equal to <?>	My_Field <> 'your_value'
is not null	My_Field IS NOT NULL
is null	My_Field IS NULL

4. Click on the criteria values box (next to the filter operator box) to activate the appropriate editor and enter the criteria values. The editor used in criteria value boxes is determined by the editor type assigned to the corresponding column.
5. Repeat step 1-4 to add another new condition.
6. Click the **<Click here to apply>** to see the result of the filtering you made.

Hint: You are allowed to save filter criteria to and load them from profiles for future use. Just simply control-click on the Filter Wizard and select **Save As**/**Load Profile**.

Query

A query is used to extract data from the database in a readable format according to the user's request. Navicat provides two powerful tools for working with the SQL queries: Query Editor for editing the query text directly and Query Builder for building queries visually. You can save your queries for setting schedule. Click to open an object list for **Query**. Or, you can simply click button in the main window.

Hint: Queries(.sql) are saved under the Settings Location.

Button	Description
Run	Run and/or Explain the query.
Stop	Stop the query.
Export	Export the result of the query.

Query Builder (Available only in Full Version)

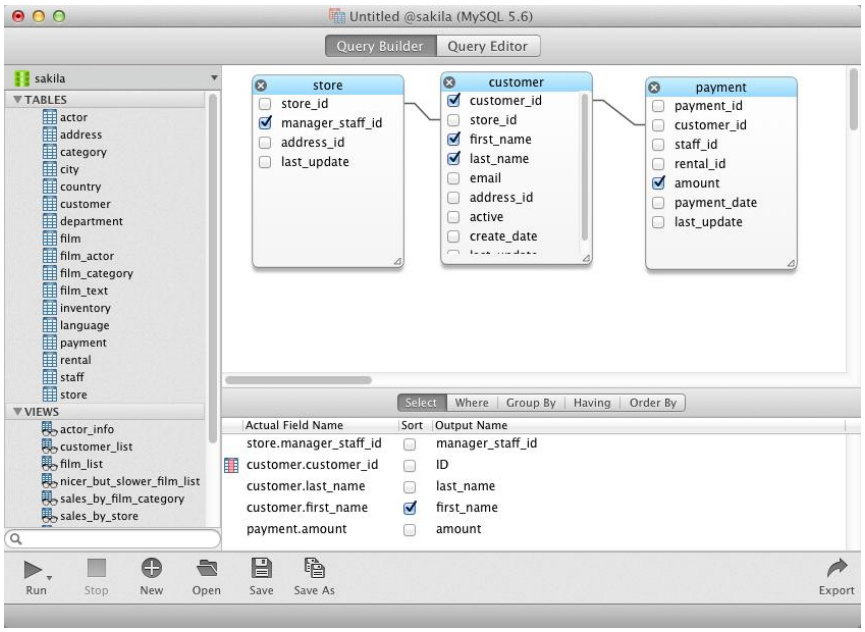
Navicat provides a useful tool called **Query Builder** for building queries visually. It allows you to create and edit queries without knowledge of SQL. The database objects are displayed in left pane. Whereas in the right pane, it is divided into two portions: the upper **Diagram Design** pane, and the lower **Criteria Selection** pane.

Note: Query Builder supports *SELECT* statement only. Use Query Editor for creating complex queries.

Drag a table from the left pane to the Diagram Design pane or double-click it to add it to query. To include a table field in the query, check the left of the field name in the Diagram Design pane. To include or deselect all the fields, control-click the object caption and choose **Select All** or **Deselect All**. To invert the selection of fields, control-click the table caption and choose **Invert Selection**.

To remove the object from the Diagram Design pane, click the cross button at the object caption.

To add the table alias, simply double-click the table name and enter the alias in the Diagram Design pane.



Setting Field Association

To associate database objects by two fields, just drag one field from the object list to another and a line will appear between the linked fields.

Hint: To remove the links of an object, control-click the object caption and choose **Unlink Other Tables**.

To change the association between the links, control-click the link and choose the properties item from the popup menu. You can change the type of Join.

Setting Output Fields

The fields you have selected in the graphical view will be displayed in the **Select** pane which allows you to set their displaying order and edit the output fields of the query.

Actual Field Name

Assumed you opened a table in the Diagram Design pane, you can click the checkbox of a field in order to add the field to Select pane.

Sort

To add a field to the sorting list, please enable the checkbox of the field in Select pane.

Output Name

Output name will be displayed in the grid of your query result. You can enter a field alias here. If you change it, you can click to hide the field in the query editor and result.

Setting Criteria

You can drag and drop a field from field box to **Where** pane. To define your own criteria, type the SQL condition statement in the pane. It will be embedded in the **WHERE** part of your query statement.

Setting Grouping

You can drag and drop a field from field box to **Group By** pane. To define your own criteria, type the SQL condition statement in the pane. The conditions will be included into the **GROUP BY** statement of the current query.

Setting Grouping Criteria

You can drag and drop a field from field box to **Having** pane. To define your own criteria, type the SQL condition statement in the pane. The conditions will be included into the **HAVING** statement of the current query.

Setting Sorting Criteria

You can set the conditions of sorting the query records. To add a field to the sorting list, please enable the checkbox **Sort** of the field in Select pane.

To set the sorting order, please change the Sort Order in **Order By** pane. You can drag and drop a field to swap with another field. The sorting order will be based on the field sequence.

To change the sorting direction, you can set the **Sort Order** to either **ASC** or **DESC**.

Query Editor

Navicat provides a useful tool called **Query Editor** for creating and executing queries. It allows you to create and edit SQL text for a query, prepare and execute selected queries.

Hint: Query text will be automatically generated while you build in Query Builder.

You are allowed to run selected portion of query, just simply control-click the highlighted query and select **Run Selected**.

You can define multiple SQL statements in one Editor window, and the editor let you run the current statement your cursor is on (place your cursor in the front of the desired statement). Just simply select **Run Statement From Here**.

Note: Select **Run Statement From Here**, the next statement will continue to run.

Editor Advanced Features

Navicat provides a wide range advanced features, such as compelling code editing capabilities, smart code-completion, sql formatting, and more.

SQL Formatting

To change the SQL statement format, simply choose from the **Format** menu -

Upper Case Keywords

Format the SQL keywords in the selected statement to upper case.

Comment

Comment the selected lines of codes.

Uncomment

Uncomment the selected lines of codes.

Beautify SQL With (Available only in Full Version)

Change the Beautify SQL settings.

Option	Description
Use tab character	Check this option to use tab character.
Tab Size	Set the tab size.
Short Brace Length	Set the length of the short brace.

Upper case keywords	Format all the SQL keywords to upper case.

Beautify SQL (Available only in Full Version)

Format the selected codes with the Beautify SQL settings.

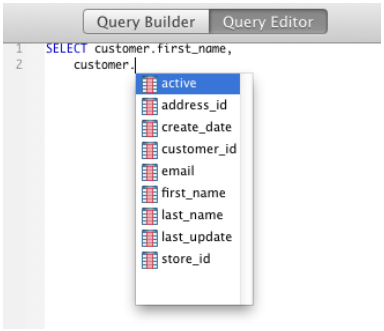
Minify SQL (Available only in Full Version)

Minify the format of the SQL in the SQL Editor.

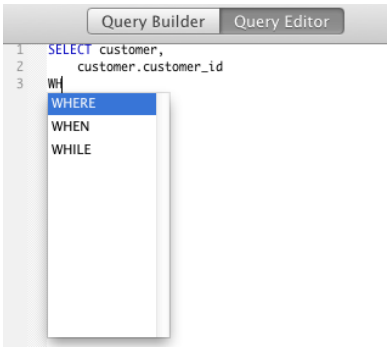
Code-Completion (Available only in Full Version)

Code-completion in Navicat displays information in drop-down lists as you type your SQL statement in the editor, it assists you with statement completion and the available properties of database objects, for example databases, tables, fields, views etc with their appropriate icons.

To activate the code-completion, just simply press **'.'** for the available properties of database object currently in the scope.



Hint: You may invoke code-completion by typing two characters or pressing ESC on your keyboard for SQL keywords.



Hint: Smart code-completion will pop-up a list of variants for the word completion automatically.

Note: Code-completion can be also applied on View, Functions/Procedures, etc.

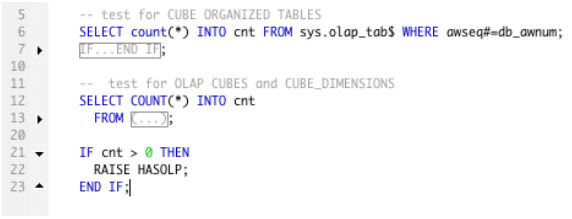
Related topic:

Preference

Code Folding

Code folding feature enables you to collapse blocks of code such that only the first line of the block appears in **Editor**.

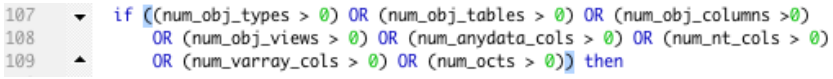
A block of code that can be folded is indicated by ▼ and ▲ to the left of the first line and the last line of the block. A vertical line extends from the icon to the bottom of the foldable code. In contrast, a folded block of code is indicated by an icon ▶ to left of the code block. You can fold the block by clicking ▼ or ▲ or expand it by clicking ▶ in **Editor**.


-- test for CUBE ORGANIZED TABLES
SELECT count(*) INTO cnt FROM sys.olap_tab$ WHERE awseq#=db_awnum;
IF...END IF;
-- test for OLAP CUBES and CUBE_DIMENSIONS
SELECT COUNT(*) INTO cnt
FROM [...];
IF cnt > 0 THEN
RAISE HASOLP;
END IF;

Brace Highlight

Navicat supports to highlight the matching brace in the editor, i.e. (), Begin...End .

Note: The cursor must be on a brace to show the highlight.


if ((num_obj_types > 0) OR (num_obj_tables > 0) OR (num_obj_columns >0)
OR (num_obj_views > 0) OR (num_anydata_cols > 0) OR (num_nt_cols > 0)
OR (num_varray_cols > 0) OR (num_octs > 0)) then

Find and Replace

Find

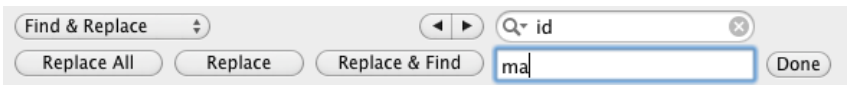
The **Find** Dialog is provided for quick searching and replacing for the text in the editor window. Just simply click **Edit** -> **Find** -> **Find** from the menu or press CMD-F and enter a search string.

Incremental searching is used here. As you type, the matched text is found and highlighted instantly. This saves your time from typing the entire text.

The search starts at the cursor's current position to the end of the file.

If you want to search a text that has been searched before, you can open the drop-down list by clicking 🔍 where a list of searching history will be displayed.

Find & Replace



Click **Replace All** to replace all occurrences automatically.

Click **Replace** to replace the first occurrence.

Click **Replace & Find** to replace the first occurrence and find the next matched string.

You can choose to perform either **String Matching** or **Regular Expression** for Find and Find & Replace. There are some additional options:

Ignore Case	Disable case sensitive search.
Wrap Around	Select to countinue searching from beginning of the query when you start the search somewhere within the query editor.
Contains	Return the objects that contain the search string.
Start With	Return the objects that start with the search string.
Whole Words	Return the objects that match the entire word of the search string.
Ends With	Return the objects that end with the search string.

Copy with Quotes

To copy the SQL statement with quotes, just simply control-click the highlighted SQL. Then, select **Copy with quotes** and choose the format.

Note: Only available in Query.

Zoom In/Zoom Out

Navicat has the ability to zoom in or zoom out the SQL in the editor. The zooming options are available from the **Edit** menu. The same effect can be achieved with keyboard shortcuts.

- Zoom In: [CMD-+] or [CMD-Mousewheel Up]
- Zoom Out: [CMD--] or [CMD-Mousewheel Down]
- Reset: [CMD-0]

Hint: Range from -4 to +5.

Note: Files are opened in different tabs will not be effected by the zoom.

Query Results

To run the query, click **Run** on the toolbar. If the query statement is correct, the query executes and, if the query statement is supposed to return data, the **Result** tab opens with the data returned by the query. If an error occurs while executing the query, execution stops, the appropriate error message is displayed.

The **Result** tab displays the result data, returned by the query, as a grid. Data can be displayed in two modes: **Grid View** and **Form View**. See Table Viewer for details.

Hint: Navicat supports to return 10 resultsets.

Note: You can choose to show the Result tab below the editor or in a new tab by selecting **Edit** -> **Show Result** -> **Below Query Editor** or **In a New Tab**.

Query Profile and Status (Available only for MySQL and MariaDB)

To show the profile and status when running the query, simply choose **Edit** -> **Show Profile and Status** and click **Run** on the toolbar.

The **Profile** tab displays the query profile: Table lock, System lock, Statistic, etc.

Note:For MySQL 5.0, support from 5.0.37 or above.

For MySQL 5.1, support from 5.1.24 or above.

The **Status** tab displays the query status: Bytes received, Bytes sent, etc.

Query Parameters

Query Builder and Query Editor both support using of parameters inside the query text. You can set query parameters to add variable values to a query each time you run it. The parameter should appear as an identifier with **$** at its beginning, quote with **[ ]**, e.g. [$any_name].

Execute the query and the **Input Parameter** Dialog is provided for you to enter the desired data you wish to search.

Debugging Oracle Query (Available only in Full Version)

To debug the Oracle query click **Debug** on the toolbar to launch the Oracle Debugger.

Enter the parameter(s) if the query has input parameter(s).

Model (Available only in Navicat Premium and Enterprise Version)

Model is a powerful tool for creating and manipulating database models. Click to open an object list for **Model**. Some of key features are listed here:

- Create and manipulate a model graphically.
- Reverse engineer a database/schema or table(s) to a model.
- Forward engineer a model to a sql file or database/schema.
- Create and edit table structures directly.

Sidebar Panel

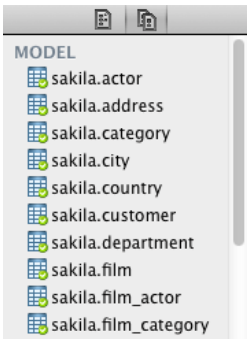
In the left and right **Sidebar** panels, all objects of your model/diagram(s), their properties and action history are listed. To show the left sidebar, simply choose **Edit** -> **Show Model Tree** in main menu. To show the right sidebar, simply choose **Edit** -> **Show Property View** in main menu.

The Sidebar consists of the following components:

- Model Tree Palette
- Diagram Tree Palette
- Properties Palette
- History Palette

Model Tree Palette

Model Tree holds all the tables in the model, including the tables used in each individual diagram.



The popup menu options of the table in model tree include:

Design Model Table

Edit an existing table in model tree.

New Model Table

Create a new table in model tree.

Delete Model Table

Delete a table in model tree.

Add to Diagram

Add the table to the active diagram.

Export SQL

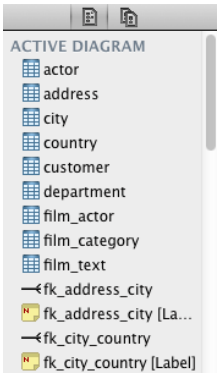
Export the table SQL to a file.

Rename

Change the name of the table.

Diagram Tree Palette

Diagram Tree holds all the objects (tables, layers, notes, images and relations) added to the active diagram.



The popup menu options of the diagram in diagram tree include:

Add Diagram

Create a new diagram.

Delete Diagram

Delete the active diagram if you have more than one diagram.

Diagram Notation

Change the notation of the diagram.

Diagram Dimensions

Change the number of pages used in the diagram.

Properties Palette

The **Properties** palette is used to display and edit the properties of diagram and its objects.

Black & white

Check this box to change the diagram color to black and white.

Bold

Check this box or press CMD-B to bold the table or relation.

Cardinality

The relation cardinality of the table.

Color

The color of the object.

Database

The database type of the diagram.

Font

The font and font size of the note.

Font Color

The font color of the note.

Name

The name of the object.

Notation

The notation of the diagram. The value for this can be Default, Simple, IDEF1X, UML or Crow's Foot.

Note Style

The style of the note. The value for this can be Note or Label.

Opacity

The transparency of the image.

Pages

The width and height of the diagram (number of papers).

Position

The number of pixels from the object to the left side (X) and the top (Y) of the canvas.

Referenced

The referenced (parent) table.

Referencing

The referencing (child) table.

Schema

The schema names of the table.

Show description

Check this box to show the linked description label of the table/layer/relation.

Show grid lines

Check this box to turn the grid on in the diagram canvas.

Show schema name

Check this box to show the schema names of the tables in the diagram.

Size

The width and height of the object.

Snap to grid

Check this box to align objects on the canvas with the grid.

Table Font

The font and font size of the table.

Visible

Check this box to show the relation lines.

History Palette

The **History** palette shows all the actions that you have taken. Simply double-click a action to restore that state.

Diagram Editor

Diagram Editor consists of a canvas and a vertical toolbar for you to design the diagram, such as adding objects, formatting diagrams and printing models, etc. Simply click the **Add Diagram** button to create a new diagram.

Create Tables

To create a new table, click the button from the diagram toolbar and click anywhere on the canvas. You can add existing table(s) from the model tree to the diagram, simply drag and drop the selected table(s) from model tree to the canvas. To show/hide the linked description label, simply check/uncheck the **Show description** option in Properties Palette.

Note: If you control-click a field, you can choose to add, insert, delete, rename field and set the field as primary key.

The popup menu options of the table object in canvas include:

Design Table

Edit the table structure in a designer, e.g. fields, indexes, foreign keys, etc. The tabs and options in the designer depend on the diagram database type you are chosen. For the settings of different tabs, see Server Objects.

Add Related Tables

Add all related tables to the selected table.

Add Field

Add fields to the existing table.

Cut

Remove the table from the diagram and put it on the clipboard.

Copy

Copy the table from the diagram to the clipboard.

Paste

Paste the content from the clipboard into the diagram.

Select All Tables

Select all the tables in the diagram.

Delete

Delete a table from the diagram or from both diagram and model.

Rename

Change the name of the table.

Color

Change the color of the table.

Size to Fit

Resize the table automatically to fit its contents.

Bring to Front

Bring table to the foreground.

Send to Back

Move table to the background.

Create Notes and Labels

Notes or Labels are typically used to help document the diagram design process. For example, to explain a grouping table objects.

To create a new note or label, click the button from the diagram toolbar and click anywhere on the canvas.

The popup menu options of the note object in canvas include:

Edit Note

Change the content of the note.

Style

Choose the style of the note: Note or Label.

Cut

Remove the note from the diagram and put it on the clipboard.

Copy

Copy the note from the diagram to the clipboard.

Paste

Paste the content from the clipboard into the diagram.

Select All Notes

Select all the notes in the diagram.

Delete

Delete a note from the diagram.

Color

Change the color of the note.

Size to Fit

Resize the note automatically to fit its contents.

Bring to Front

Bring note to the foreground.

Send to Back

Move note to the background.

Create Images

To create a new image, click the button from the diagram toolbar and click anywhere on the canvas.

The popup menu options of the image object in canvas include:

Reset Size

Reset the size of the image to its original size.

Reset Aspect Ratio

Maintain image original width to height ratio.

Cut

Remove the image from the diagram and put it on the clipboard.

Copy

Copy the image from the diagram to the clipboard.

Paste

Paste the content from the clipboard into the diagram.

Select All Images

Select all the images in the diagram.

Delete

Delete an image from the diagram.

Bring to Front

Bring image to the foreground.

Send to Back

Move image to the background.

Create Layers

Layers are used to help organize objects (e.g. tables, notes, images, etc) on the canvas. You can add all related objects to the same layer. For example, you may choose to add all your sales related tables to one layer. To create a new layer, click the button from the diagram toolbar and click anywhere on the canvas. To show/hide the linked description label, simply check/uncheck the **Show description** option in Properties Palette.

The popup menu options of the layer object in canvas include:

Cut

Remove the layer from the diagram and put it on the clipboard.

Copy

Copy the layer from the diagram to the clipboard.

Paste

Paste the content from the clipboard into the diagram.

Select All Layers

Select all the layers in the diagram.

Delete

Delete a layer from the diagram.

Color

Change the color of the layer.

Size to Fit

Resize the layer automatically to fit its contents.

Bring to Front

Bring layer to the foreground.

Send to Back

Move layer to the background.

Create Relations

To add a relation, click the button from the diagram toolbar and drag and drop a field from one table to another. To show/hide the linked description label, simply check/uncheck the **Show description** option in Properties Palette.

The popup menu options of the relation object in canvas include:

Design Relation

Edit the relation in a designer. The options in the designer depend on the diagram database type you are chosen. For the settings, see Server Objects.

Cardinality on table_name1

Set the cardinality on table_name1: None, One and Only One, Many, One or Many, Zero or One, Zero or Many.

Cardinality on table_name2

Set the cardinality on table_name2: None, One and Only One, Many, One or Many, Zero or One, Zero or Many.

Add Vertex

Add a vertex on a relation connecter.

Delete Vertex

Delete a vertex on a relation connecter.

Delete All Vertices

Delete all vertices on a relation connecter.

Paste

Paste the content from the clipboard into the diagram.

Select All Relations

Select all the relations in the diagram.

Delete from Diagram and Model

Delete a relation from the diagram and model.

Color

Change the color of the relation.

Format Diagram

Show Grid Lines

To turn the grid on in the diagram canvas, choose **Edit** -> **Show Grid Lines** from the menu.

Snap to Grid

To align objects on the canvas with the grid, choose **Edit** -> **Snap to Grid** from the menu.

Change Database Type

To change the database type of the diagram, choose **Edit** -> **Database** and select the type from the menu. Choose **Generic** if you do not decide the database type.

Change Diagram Notation

To change the notation of the diagram, choose **Edit** -> **Diagram Notation** and select the notation from the menu.

Default	The default notation style used in Navicat.
Simple	A simple notation style. The table objects will only show the name.
IDEF1X	The ICAM DEFinition language information modeling method.
UML	Universal Modeling Language style.
Crow's Foot	Crow's Foot notation style.
Black and White	Change the color of the diagram to black and white.
Schema Name	Show the schema names of the tables in the diagram.

Change Diagram Dimensions

To change the number of pages used in the diagram, choose **Edit** -> **Diagram Dimensions** from the menu and set the Width and Height.

Align Objects

To align objects on the canvas, select more than one object (table/note/image), then control-click and choose **Alignment** -> **Align Left**/**Align Center**/**Align Right**/**Align Top**/**Align Middle**/**Align Bottom**.

Change the Objects Distribution

To distribute objects on the canvas, select more than one object (table/note/image), then control-click and choose **Distribute** -> **Horizontal**/**Vertical**.

Change Page Setup

To change paper size, orientation and margins, choose **File** -> **Page Setup**.

Apply Auto Layout

To automatically arrange objects on the canvas, click **Auto Layout** from the toolbar. To change the Auto Layout format, simply choose **Edit** -> **Auto Layout with** from the menu and set the following options:

Option	Description
Space Between Objects	The distance between the objects in the diagram.
Number of Trials	The quality of the auto layout output.
Auto Dimension	Choose the suitable diagram dimension automatically.
Tables resize to Fit	Resize the table to fit its content automatically.

Zoom In/Zoom Out

To zoom in or zoom out the selected area of the diagram, adjust the slider of the **Navigator**. Same effect can be achieved with keyboard shortcuts:

Zoom In: [CMD-+] or [CMD-Mousewheel Up]

Zoom out: [CMD--] or [CMD-Mousewheel Down]

Preview and Print Model

Simply click **Print** to send your diagram directly to the printer. You can set the printer option in the popup window.

You can also choose **File** -> **Print PDF**/**Print PNG**/**Print SVG** to create a PDF/PNG/SVG file of your diagram.

Reverse Engineering

Reverse Engineering is one of the key features of Model. This feature allows you to load already existing database structures to create new diagrams. It supports to import MySQL, PostgreSQL, Oracle, SQLite, SQL Server or MariaDB databases/tables.

Navicat provides a step-by-step wizard for you to complete the task:

1. Select **File** -> **Import from Database**.
2. Select a connection.
3. Choose databases, schemas or tables you want to import.
4. Choose import options: include indexes, foreign keys, uniques and/or comments
5. Click **Start**.

You can also simply create a new model using reverse engineering in the Navicat main window. Control-click an opened database/schema or table(s) and select **Reverse Database to Model** or **Reverse Tables to Model** from the popup menu.

Script Generation

After finishing your model, you can save table structures and relations from the model into a script file. The **Export SQL** feature generates a SQL file for the script. Select **File** -> **Export SQL**.

General Settings for Export SQL

Export to File

Set the output file name and location.

Select tables to export

Choose tables in current model you wish to export.

Advanced Settings for Export SQL

The following options depend on the diagram database type you are chosen: Generic, MySQL, Oracle, PostgreSQL, SQLite, SQL Server and MariaDB.

Server Version

Select server version for the SQL file.

Include Schema Name

Include entered schema name in file with this option is on. Otherwise, only table name is included in SQL statements.

Include Drop SQL

Include drop table SQL statements in file with this option in on.

Include Drop with CASCADE

Include drop table SQL statements with cascade option in file with this option in on.

Include Primary Keys

Include primary keys in file with this option is on.

Include Foreign Keys

Include foreign keys in file with this option is on.

Include Uniques

Include uniques in file with this option is on.

Include Indexes

Include indexes in file with this option is on.

Include Character Set

Include table and field character set in file with this option is on.

Include Auto Increment Value

Include table auto increment values in file with this option is on.

Include Collation

Include table collation in file with this option is on.

Forward Engineering

Forward engineering is one of the key features of Model. This feature allows you to compare the model and existing schemas/tables, states the differences between their structures, and offer synchronizing the structures in model to the target connection.

Navicat provides a step-by-step wizard for you to complete the task.

1. Select **File** -> **Synchronize to Database**.
2. Select the synchronization type.
3. Select the target connection from existing connections.
4. Select the source schemas/tables.
5. Edit synchronization properties.
6. Click **Compare** to generate a set of scripts that show the differences between source and target tables.
7. Select the scripts you want to run.
8. Click **Run** button.

Selecting Synchronization Type

Sync with selected schemas

Set the synchronization to work on all tables in schemas.

Sync with selected tables

Set the synchronization to work on tables only.

Selecting Target Connection

Chooses target connection and database from existing connections.

Selecting Schemas/Tables

Choose one or more schemas/tables in model to compare to the target schemas/tables in this step. If tables in model are from existing schemas, you can select the existing schemas. Otherwise, select a target schema name for the source model tables to compare to.

Selecting Synchronize Options

The following options depend on the diagram database type you are chosen: MySQL, Oracle, PostgreSQL, SQLite, SQL Server and MariaDB.

Compare in case sensitive

Check this option if you want to compare table identifier with case sensitive option.

Compare primary keys

Check this option if you want to compare table primary keys.

Compare foreign keys

Check this option if you want to compare table foreign keys.

Compare indexes

Check this option if you want to compare indexes.

Compare character set

Check this option if you want to compare character set of tables.

Compare uniques

Check this option if you want to compare uniques.

Compare auto increment value

Check this option if you want to compare table auto increment values.

Compare collation

Check this option if you want to compare collation of tables.

SQL for objects to be created

Check this option to include all related SQL statements if new objects will be created in the target database.

SQL for objects to be changed

Check this option to include all related SQL statements if objects will be changed in the target database.

SQL for objects to be dropped

Check this option to include all related SQL statements if objects will be dropped from the target database.

Continue on error

Ignore errors that are encountered during the synchronization process.

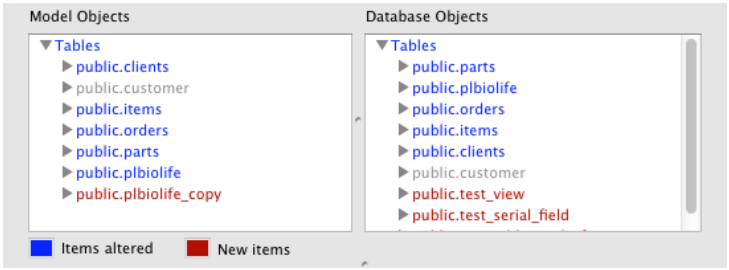
Viewing Comparison Result

Model Objects/Database Objects

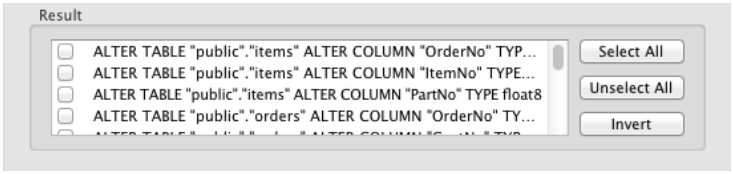
The tree view shows the differences between model and database tables after the comparison of their structures, providing with the detailed SQL statements shown in the **Result** list.

The red item represents the non-existence for the other database/schema.

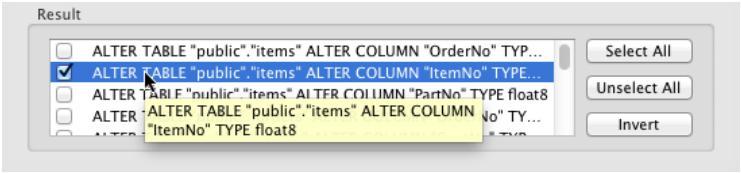
The blue item represents the existence for the other database/schema, but different definition detected.



All the scripts are checked in the **Result** list by default.


Result
ALTER TABLE "public"."items" ALTER COLUMN "OrderNo" TYP...
ALTER TABLE "public"."items" ALTER COLUMN "ItemNo" TYPE...
ALTER TABLE "public"."items" ALTER COLUMN "PartNo" TYPE float8
ALTER TABLE "public"."orders" ALTER COLUMN "OrderNo" TY...
Select All
Unselect All
Invert

To view the full SQL statement, move mouse cursor hovers over a statement.


Result
ALTER TABLE "public"."items" ALTER COLUMN "OrderNo" TYP...
ALTER TABLE "public"."items" ALTER COLUMN "ItemNo" TYPE...
ALTER TABLE "public"."items" ALTER COLUMN "PartNo" TYPE float8
ALTER TABLE "public"."items" ALTER COLUMN "ItemNo" TYPE float8
Select All
Unselect All
Invert

Click **Continue** button to execute the selected query.

Model Hints and Tips

Navicat provides some useful hints to work on the model more effectively.

Action	Description
Locate Object in Diagram Editor	- Object selected in Diagram Tree will be highlighted in Diagram Editor.
Delete Table from Model Tree Palette	- Select table in Diagram Editor and press DELETE.
Open Table Designer	- Double-click a table in Diagram Editor.
Add table from Navicat Main	- Drag table from Navicat main window and drop to the Diagram Editor.
Get Table Structure (SQL Statement)	- Select and copy the table in Diagram Editor, and paste it to text editor, e.g. TextEdit.
Design Field without Table Designer	- Select and click the table name and press DOWN ARROW to add/edit fields. Navicat will predict field types according to field names you entered. **INTEGER/int/int4/NUMBER** - suffix "id", "no" (if it is the first column, it will be predicted as a primary key) - suffix "num" - "qty", "number"

	- exactly "age", "count" **DECIMAL(10,2)/decimal(10,2)/NUMBER/REAL/money** - suffix "price", "cost", "salary" **FLOAT/double/float8/NUMBER/REAL/float** - "size", "height", "width", "length", "weight", "speed", "distance" **DATE/datetime/date/TEXT/datetime2** - "date", "time" **VARCHAR(255)/varchar(255)/VARCHAR2(255)/TEXT** - other field names Enter * before the field name to recognize as primary key. e.g. *itemNo:int. Enter : between field name and field type to custom field type, e.g. itemName:varchar(255).
Reorder Field	- Select table in Diagram Editor, then press and hold the SHIFT key. Use to drag the field to a desired location.
Delete Field	- Select table in Diagram Editor, then press and hold the SHIFT key. Use to drag the desired field out of the table.
Add Vertex to Relation	- Select relation in Diagram Editor, then press and hold the SHIFT key. Click on the relation to add vertex.
Delete Vertex on Relation	- Select relation in Diagram Editor, then press and hold the SHIFT key. Click on the vertex.
Delete Relation from Model	- Select relation in Diagram Editor and press SHIFT+DELETE.
Switch to Hand Mode	- Press and hold the SPACE key, then move the diagram.

Advanced Tools

Navicat provides a number of powerful tools for working with data, which includes **Import Wizard**, **Export Wizard**, **Dump SQL File**, **Execute SQL File** and more.

Import Wizard

Import Wizard allows you to import data to tables from CSV, TXT, XML, DBF and more. You can save your settings as a profile for setting schedule.

Note: Navicat Essentials version only supports to import text-based files, such as TXT, CSV, HTML, XML and JSON file.

To open the Import Wizard, click **Import** from the table object list toolbar.

Hint: You can drag a supported file to a table or a database/schema in the connection tree. Navicat will popup the **Import Wizard** window. If existing table is highlighted, Navicat will import the file to the highlighted table. Otherwise, import the file to a new table.

Setting Import File Format

Select one of the available import types for the source file.

Setting Source File Name

Browse the source file or provide the file path directly. You can select more than one file to import. Select the **Encoding** for the source file.

ODBC

Setting up an ODBC Data Source Connection

1. Install suitable ODBC Administrator and the corresponding driver for file.
2. In Applications, select **Utilities**.
3. Select **ODBC Administrator** or click **ODBC Administrator** button in second step of the Import Wizard in Navicat.
4. Click **Add** button in **User DSN** tab.
5. Select the appropriate ODBC driver and click **OK** button.
6. Enter required information.
7. Click **Finish** button to see your ODBC Driver in the list.

Note: You can consult with the driver provider about how to setup the DSN.

Connecting to ODBC Data Source in Navicat

1. Click ... in **Import from** in second step of the Import Wizard.
2. Choose the data source from the **Connection** drop-down list and provide valid username and password.

3. All available tables will be included in the list in next step if connection is success.

Setting Delimiter

TXT, CSV

Define **Field Delimiter**, **Record Delimiter** and **Text Qualifier** for file.

XML

Select the **Source File** and define tag to identify table row.

Consider tag's attributes as table's field

For example:

<row age="17">
<id>1</id>
<name>sze</name>
</row>

With this option is on, Navicat will recognizes "age" as a table field together with "id" and "name", otherwise, only "id" and "name" will be imported as table fields.

Note: Navicat does not support multiple level of XML file.

Excel, Access

Worksheets will be shown in the list.

ODBC

The **Add Query**, **Delete Query** and **Modify Query** buttons open the **Add Query** dialog where you can construct query to import only certain rows from your source tables. In other words, import only rows that satisfy the criteria set by you. Tables or queries will be shown in the list.

Setting Additional Options

First Row

First row indicates which row should Navicat start reading the actual data.

Last Row

Last row indicates which row should Navicat stop reading the actual data.

Field Name Row

Field name row indicates which row should Navicat recognize as Column Title.

Note: If no column title is defined for the file, please enter **1** for First row and **0** for Field name row.

Date Format, **Date Delimiter**, **Time Delimiter**

Define the formats of date and time.

Four Digit Years

Check this option to display four digits for years.

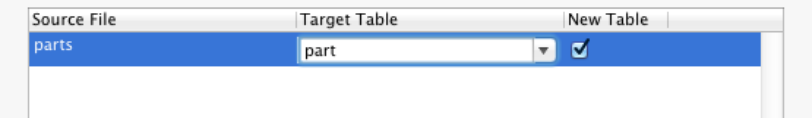
Decimal Symbol, **Thousand Separator**

Define the format of number.

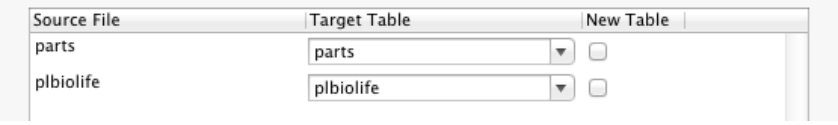
Setting Target Table

You are allowed to define a new table name or choose to import into an existing table from the drop-down list.

Note: If you type a new table name in **Target Table**, the box in **New Table** will be checked automatically.



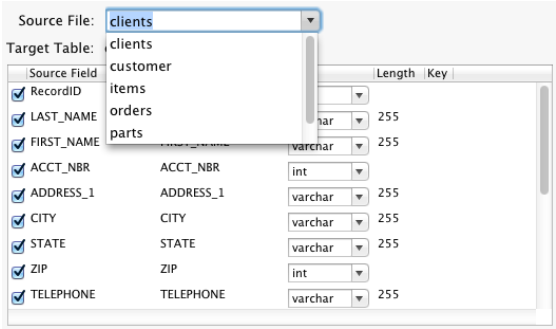
For importing multiple tables, all tables will be shown in the list.



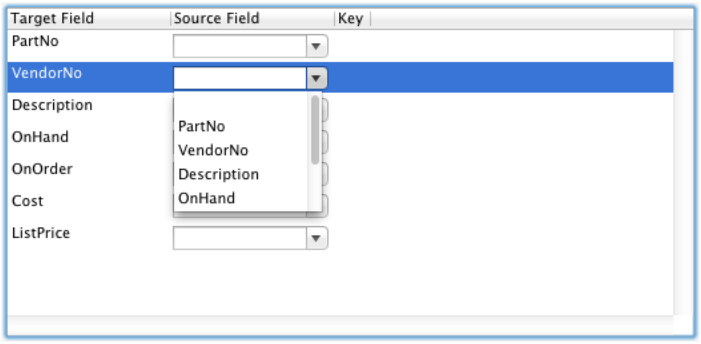
Adjusting Field Structures and Mapping Fields

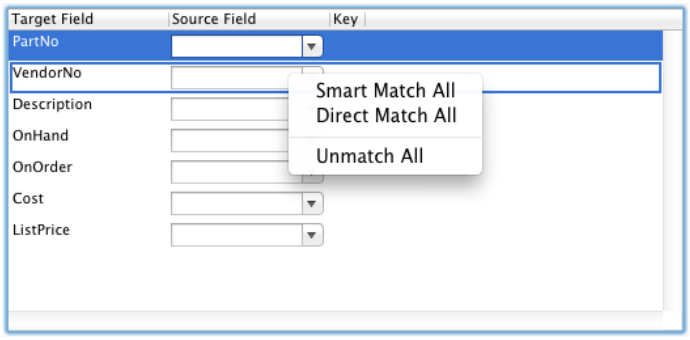
Navicat will make assumption on the field types and length in the source table. You are allowed to choose desired type from the drop-down list.

Hint: For importing multiple tables, select other tables from the **Source File** drop-down list.



If you are importing data into existing table(s), you might need to map the source field names manually to the destination table or right-click and select **Smart Match All**, **Direct Match All** and **Unmatch All** from the popup menu for quick mapping.



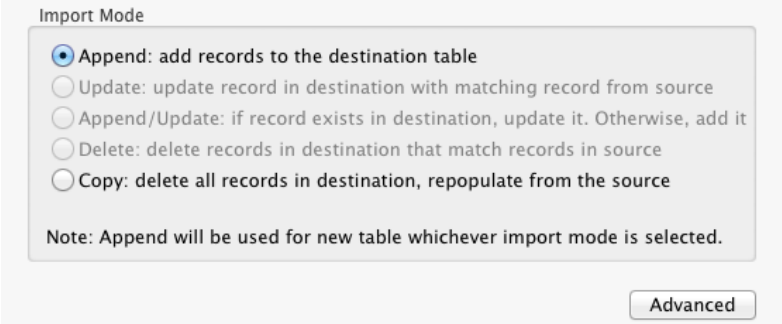


If you are importing via ODBC, the **Condition Query** button opens the **WHERE** dialog where you can specify a *WHERE* clause to import only certain rows from your source tables. In other words, import only rows that satisfy the criteria set by you.

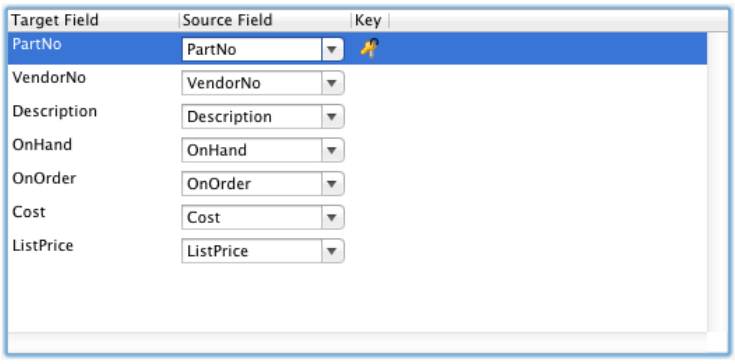
Hint: Do not include the word *WHERE* in the clause.

Selecting Import Mode

Select the import mode that defines how the data being imported.


Import Mode
Append: add records to the destination table
Update: update record in destination with matching record from source
Append/Update: if record exists in destination, update it. Otherwise, add it
Delete: delete records in destination that match records in source
Copy: delete all records in destination, repopulate from the source
Note: Append will be used for new table whichever import mode is selected.
Advanced

Hint: To activate the remaining options, you must enable Primary Key in previous step.



Click **Advanced** button for more settings:

The following options depend on the database type you are chosen: MySQL, Oracle, PostgreSQL, SQLite, SQL Server and MariaDB.

Use extended insert statements

Insert records using extended insert syntax.

Example:

INSERT INTO `users` VALUES ('1', 'Peter McKindsy', '23'), ('2', 'Johnson Ryne', '56'), ('0', 'Katherine', '23');

Run multiple insert statements

Check this option if you want to run multiple insert statements in each execution.

Use empty string as NULL

Import **NULL** value if the source data field contains empty string.

Use foreign key constraint

Add foreign key if there is foreign key relations between tables.

Continue on error

Ignore errors that are encountered during the import process.

Saving and Confirming Import

Click **Start** button to start the import process. You can view the running process indicating success or failure.

Hint: Click **Save** button to save your settings as a profile for setting schedule.

You can click **View log** button to view the log file.

Export Wizard

Export Wizard allows you to export data from tables, views, or query results to any available formats. You can save your settings as a profile for setting schedule.

Note: Navicat Essentials version only supports to export text-based files, such as TXT, CSV, XML and JSON file.

To open the Export Wizard, select an existing table/view/query and click **Export** from the object list toolbar.

Setting Export File Format

Select one of the available export formats for the target file.

Setting Destination File Name

Set exported file name and location. The file extension in the **File Name** text box changes according to the selected export type in first step. The highlighted table is checked automatically. If you are exporting selected tables into the same target file, set them with the same file name. When the file format is Excel file, you can control-click and select **Export Selected to Same File** from the popup menu.

Note: For exporting query result, ensure the query is saved before running the Export Wizard. Otherwise, no source table displayed in here.

Encoding

Select the encoding for the exported file.

Add timestamp

Check this option if you want your file name specifies the timestamp of the export is run. Select the date/time format from the drop-down list.

Selecting Fields for Export

Select table fields for export. All the fields are selected in the list by default. If you want to omit some fields to be exported, uncheck the box **All Fields** first and then uncheck those fields in the list.

Note: For exporting query result, the wizard will skip this step.

Setting Additional Options

The following options depend on the file format chose in first step.

Include column titles

Field names will be included into the exported file if this option is on.

Blank if zero

Output zero if the content of field is blank.

Append on output file(s)

If you set exporting multiple tables to the same target file in second step, check this option to append records in the exported file.

Continue on error

Ignore errors that are encountered during the export process.

Define **Record Delimiter**, **Field Delimiter** and **Text Qualifier** for file.

Date Format, **Date Delimiter**, **Time Delimiter**

Define the formats of date and time.

Decimal Symbol

Define the format of decimal number.

Saving and Confirming Export

Click **Start** button to start the export process. You can view the running process indicating success or failure.

Hint: Click **Save** button to save your settings as a profile for setting schedule.

Data Transfer (Available only in Full Version)

Navicat allows you to transfer database objects from one database and/or schema to another, or to a sql file. The target database and/or schema can be on the same server as the source or on another server. You can save your settings as a profile for setting schedule. Select **Tools** -> **Data Transfer** from the main menu or press CMD-SHIFT-T.

Hint: You can drag tables to a database/schema in the connection tree. If the target database/schema is within the same connection, Navicat will copy the table directly. Otherwise, Navicat will popup the **Data Transfer** window.

To open a saved profile, select the profile and click **Load** button or double-click it in **Profile** tab.

General Settings for Data Transfer

Source

Define connection, database and/or schema for the source.

All the database objects are selected in the **Objects** list by default. If you do not want some database objects to be transferred, uncheck them.

With this option is on, only the checked database objects will be transferred. However, if you add any new database objects in the source database and/or schema after you create your data transfer profile, the newly added database objects will not be transferred unless you manually modify the **Objects** list.

Choose this option if you wish all the database objects being transferred to the target database/schema, all newly added database objects will also be transferred without amending the data transfer profile.

Target

Connection

Transfer your selected database objects directly to a connection, database and/or schema.

File

Transfer your selected database objects directly to a text file. You can select different **Server Type**, **Version** and **Encoding** for the file.

Advanced Settings for Same Server Type Data Transfer

In this tab, you can choose the advanced settings for transferring between same server type or between MySQL and MariaDB.

The following options depend on the database type you are chosen: MySQL, Oracle, PostgreSQL, SQLite, SQL Server and MariaDB.

Create tables

Create tables in the target database and/or schema with this option is on.

Suppose this option is unchecked and tables already exist in the target database/schema, then all data will be appended to the destination tables.

Include indexes

Include indexes in the table with this option is on.

Include foreign key constraints

Include foreign keys in the table with this option is on.

Include engine/table type

Include table type with this option is on.

Include character set

Include character set in the table with this option is on.

Include auto increment

Include auto increment in the table with this option is on.

Include unique constraints

Include uniques in the table with this option is on.

Include rules

Include rules in the table with this option is on.

Include check constraints

Include checks in the table with this option is on.

Include storage

Include storage in the table with this option is on.

Include exclude constraints

Include exclusion constraints in the table with this option is on.

Include triggers

Include triggers in the table with this option is on.

Convert object name to

Check this option if you require convert object names to **Lower case** or **Upper case** during the process.

Insert records

Check this option if you require all records to be transferred to the destination database and/or schema.

Lock target tables

Lock the tables in the target database and/or schema during the data transfer process.

Use transaction

Check this option if you use transaction during the data transfer process.

Use complete insert statements

Insert records using complete insert syntax.

Example:

INSERT INTO `users` (`ID Number`, `User Name`, `User Age`) VALUES ('1', 'Peter McKindsy', '23');

INSERT INTO `users` (`ID Number`, `User Name`, `User Age`) VALUES ('2', 'Johnson Ryne', '56');

INSERT INTO `users` (`ID Number`, `User Name`, `User Age`) VALUES ('0', 'katherine', '23');

Use extended insert statements

Insert records using extended insert syntax.

Example:

INSERT INTO `users` VALUES ('1', 'Peter McKindsy', '23'), ('2', 'Johnson Ryne', '56'), ('0', 'Katherine', '23');

Use delayed insert statements

Insert records using *DELAYED* insert SQL statements.

Example:

INSERT DELAYED INTO `users` VALUES ('1', 'Peter McKindsy', '23');

INSERT DELAYED INTO `users` VALUES ('2', 'Johnson Ryne', '56');

INSERT DELAYED INTO `users` VALUES ('0', 'katherine', '23');

Run multiple insert statements

Check this option if you want to run multiple insert statements in each execution, which will make the data transfer process faster.

Use hexadecimal format for BLOB

Insert BLOB data as hexadecimal format.

Continue on error

Ignore errors that are encountered during the transfer process.

Lock source tables

Lock the tables in the source database and/or schema so that any update on the table is not allowed once the data transfer is triggered off.

Use single transaction (InnoDB only)

If a table uses InnoDB storage engine, with this option is on, Navicat uses transaction before the backup process starts.

Drop target objects before create

Check this option if database objects already exist in the target database and/or schema, the existing objects will be deleted once the data transfer starts.

Drop with cascade

Check this option to drop objects with cascade.

Create target database if not exist

Create a new database if the database specified in target server does not exist.

Create target database/schema if not exist

Create a new database/schema if the database/schema specified in target server does not exist.

Use DDL from SHOW CREATE statements

If this option is on, DDL will be used from show create table.

Include schema name in statements

Include schema name with this option is on. Otherwise, only table name is included in SQL statements.

Use DDL from sqlite_master

If this option is on, DDL will be used from the *SQLITE_MASTER* table.

Advanced Settings for Cross Server Data Transfer (Available only in Navicat Premium)

Navicat Premium supports transferring table with data across different server types, e.g. from MySQL to Oracle. If you are transferring between MySQL and MariaDB, you can refer to Advanced Settings for Same Server Type Data Transfer.

The following options depend on the database type you are chosen: MySQL, Oracle, PostgreSQL, SQLite, SQL Server and MariaDB.

Create tables

Create tables in the target database and/or schema with this option is on.

Suppose this option is unchecked and tables already exist in the target database/schema, then all data will be appended to the destination tables.

Include indexes

Include indexes in the table with this option is on.

Include foreign key constraints

Include foreign keys in the table with this option is on.

Convert object name to

Check this option if you require convert object names to **Lower case** or **Upper case** during the process.

Insert records

Check this option if you require all records to be transferred to the destination database and/or schema.

Lock target tables

Lock the tables in the target database and/or schema during the data transfer process.

Use transaction

Check this option if you use transaction during the data transfer process.

Use complete insert statements

Insert records using complete insert syntax.

Example:

INSERT INTO `users` (`ID Number`, `User Name`, `User Age`) VALUES ('1', 'Peter McKindsy', '23');

INSERT INTO `users` (`ID Number`, `User Name`, `User Age`) VALUES ('2', 'Johnson Ryne', '56');

INSERT INTO `users` (`ID Number`, `User Name`, `User Age`) VALUES ('0', 'katherine', '23');

Use extended insert statements

Insert records using extended insert syntax.

Example:

INSERT INTO `users` VALUES ('1', 'Peter McKindsy', '23'), ('2', 'Johnson Ryne', '56'), ('0', 'Katherine', '23');

Use delayed insert statements

Insert records using *DELAYED* insert SQL statements.

Example:

INSERT DELAYED INTO `users` VALUES ('1', 'Peter McKindsy', '23');

INSERT DELAYED INTO `users` VALUES ('2', 'Johnson Ryne', '56');

INSERT DELAYED INTO `users` VALUES ('0', 'katherine', '23');

Run multiple insert statements

Check this option if you want to run multiple insert statements in each execution, which will make the data transfer process faster.

Use hexadecimal format for BLOB

Insert BLOB data as hexadecimal format.

Continue on error

Ignore errors that are encountered during the transfer process.

Lock source tables

Lock the tables in the source database and/or schema so that any update on the table is not allowed once the data transfer is triggered off.

Use single transaction (InnoDB only)

If a table uses InnoDB storage engine, with this option is on, Navicat uses transaction before the backup process starts.

Drop target objects before create

Check this option if database objects already exist in the target database and/or schema, the existing objects will be deleted once the data transfer starts.

Drop with cascade

Check this option to drop objects with cascade.

Create target database if not exist

Create a new database if the database specified in target server does not exist.

Create target database/schema if not exist

Create a new database/schema if the database/schema specified in target server does not exist.

Data Synchronization (Available only in Full Version)

Navicat allows you to transfer data from one database and/or schema to another with detailed analytical process. In other words, Navicat provides the ability for data in different databases and/or schemas to be kept up-to-date so that each repository contains the same information. You are not only authorized to rollback the transferring process, but also insert, delete and update records to the destination. You can save your settings as a profile for setting schedule. Select **Tools** -> **Data Synchronization** from the main menu.

All tables must contain primary keys and all table structures must be identical between the source and target. You could apply Structure Synchronization before Data Synchronization.

To open a saved profile, select the profile and click **Load** button or double-click it in **Profile** tab.

Note: SQL Server 2000 does not support.

For Oracle server, BLOB, CLOB, NCLOB, LONG and LONG RAW data are skipped during the data synchronization process.

TIMESTAMP primary key cannot synchronize (insert, update) with Database Link to 9i server. RAW primary key cannot synchronize (insert, update, delete) with Database Link to any server, without error.

Navicat Premium and Navicat for MySQL support synchronize between MySQL and MariaDB.

General Settings for Data Synchronization

Source/Target

Define connection, database and/or schema for the source and target.

Note: For Oracle server, you need to create Public/Private Database Link to the target Oracle database before.

Source Table/Target Table

Only tables which contain identical table names between the source and target are mapped in the list by default. If you do not want some tables to be synchronized, disable them manually from the drop-down list.

Hint: You can preview the outcome before execution.

Advanced Settings for Data Synchronization

The following options depend on the database type you are chosen: MySQL, PostgreSQL, Oracle, SQLite, SQL Server and MariaDB.

Insert records, **Delete records**, **Update records**

Check these options to performing such actions to the target when data are synchronized.

Save synchronization queries to file

Check this option and select the path to save all queries in a text file.

Commit transaction in case of error

Rollback all data when error occurs.

Ensure that table structures match (recommended)

Check this option if you want to synchronize data only if the table structure of two tables are matched.

Disable foreign key checks

Check this option if you want to disable foreign key check when synchronizing data.

Structure Synchronization (Available only in Full Version)

Navicat allows you to compare and modify the table structures with detailed analytical process. In other words, Navicat compares tables between two databases and/or schemas and states the differential in structure. Select **Tools** -> **Structure Synchronization** from the main menu.

To open a saved profile, select the profile and click **Load** button or double-click it in **Profile** tab.

Note: Available only for MySQL, Oracle, PostgreSQL, SQL Server and MariaDB. Navicat Premium and Navicat for MySQL support synchronize between MySQL and MariaDB.

General Settings for Structure Synchronization

The following options depend on the database type you are chosen: MySQL, Oracle, PostgreSQL, SQL Server and MariaDB.

Source/Target

Define connection, database and/or schema for the source and target.

Compare tables

Check this option if you want to compare tables between the source and target.

Compare primary keys

Check this option if you want to compare table primary keys.

Compare foreign keys

Check this option if you want to compare table foreign keys.

Compare character set

Check this option if you want to compare character set of tables.

Compare auto increment value

Check this option if you want to compare auto increment values of tables.

Compare uniques

Check this option if you want to compare uniques.

Compare checks

Check this option if you want to compare checks.

Compare rules

Check this option if you want to compare rules.

Compare excludes

Check this option if you want to compare exclude constraints.

Compare collation

Check this option if you want to compare collation of tables.

Compare identity last value

Check this option if you want to compare table identity last values.

Compare views

Check this option if you want to compare views.

Compare stored routines

Check this option if you want to compare stored routines.

Compare events

Check this option if you want to compare events.

Compare functions

Check this option if you want to compare functions.

Compare procedures and functions

Check this option if you want to compare procedures and functions.

Compare indexes

Check this option if you want to compare indexes.

Compare sequences

Check this option if you want to compare sequences.

Compare triggers

Check this option if you want to compare triggers.

Compare tablespace and physical attributes

Check this option if you want to compare tablespace and physical attributes.

Compare storage

Check this option if you want to compare storage.

SQL for objects to be created

Check this option to include all related SQL statements if new database objects will be created in the target.

SQL for objects to be changed

Check this option to include all related SQL statements if database objects will be changed in the target.

SQL for objects to be dropped

Check this option to include all related SQL statements if database objects will be dropped from the target.

Drop with cascade

Check this option if you want to cascade to drop the dependent database objects.

Compare after execution

Compare tables after the synchronization is executed.

Continue on error

Ignore errors that are encountered during the synchronization process.

Create inheriting parent

Create tables of inheriting parents during the synchronization process.

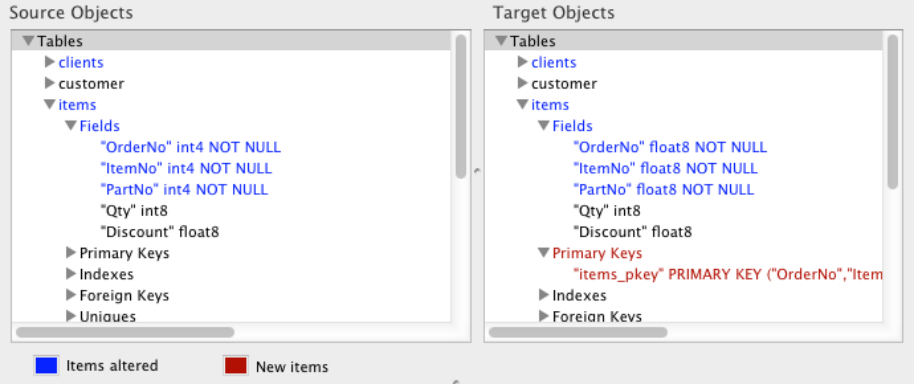
Structure Synchronization Result

Source Objects/Target Objects

The tree view shows the differences between the source and target database and/or schema after the comparison of their structures, providing with detailed SQL statements shown in the **Result** list.

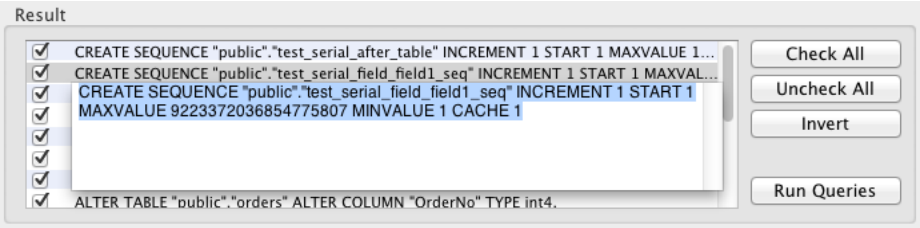
The red item represents the non-existence for the other database/schema.

The blue item represents the existence for the other database/schema, but different definition detected.



All the scripts are checked in the **Result** list by default.

To view the full SQL statements, simply click on the scripts.


Result
CREATE SEQUENCE "public"."test_serial_after_table" INCREMENT 1 START 1 MAXVALUE 1...
CREATE SEQUENCE "public"."test_serial_field_field1_seq" INCREMENT 1 START 1 MAXVAL...
CREATE SEQUENCE "public"."test_serial_field_field1_seq" INCREMENT 1 START 1
MAXVALUE 9223372036854775807 MINVALUE 1 CACHE 1
ALTER TABLE "public"."orders" ALTER COLUMN "OrderNo" TYPE int4;
Check All
Uncheck All
Invert
Run Queries

Click **Run Queries** button to execute the selected query.

Backup/Restore (Available only in Full Version)

A secure and reliable server is closely related to performing regular backups, as failures will probably occur sometimes - caused by attacks, hardware failure, human error, power outages, etc.

Navicat allows you to backup/restore database objects for your database. You can save your settings as a profile for setting schedule.

Click to open an object list for **Backup**.

Hint: Backup files are stored under Settings Location.

Note: Available only for MySQL, PostgreSQL, SQLite and MariaDB. To backup Oracle, see Oracle Data Pump. To backup SQL Server, see SQL Server Backup/Restore.

Backup

General Properties

You can view information of the backup file.

Object Selection

Choose database objects you wish to backup.

Advanced Properties

The following options depend on the database type you are chosen: MySQL, PostgreSQL, SQLite and MariaDB.

Lock all tables

Lock all objects while backup is being processed.

Use single transaction (InnoDB only)

If a table uses InnoDB storage engine, with this option is on, Navicat uses transaction before the backup process starts.

Use specified file name

Define your file name for backup. Otherwise, your backup file will be named as **"2007-05-10 17:38:20"** for example.

Restore

Restore feature will firstly drop the selected objects of the database, then recreate the new objects according to your backup. Finally, inserting the data.

To restore a backup to an existing database, open a database and select an existing backup file. Click **Open** from the backup object list toolbar.

To restore a backup to a new database, create and open a new database and click **Open** from the backup object list toolbar. Browse the backup file.

Note: You must have Create, Drop and Insert privileges (MySQL/MariaDB or PostgreSQL) to run the restore.

General Properties

Show information of the backup file.

Extract SQL

Extract SQL allows to extract SQL into a SQL file from your backup file.

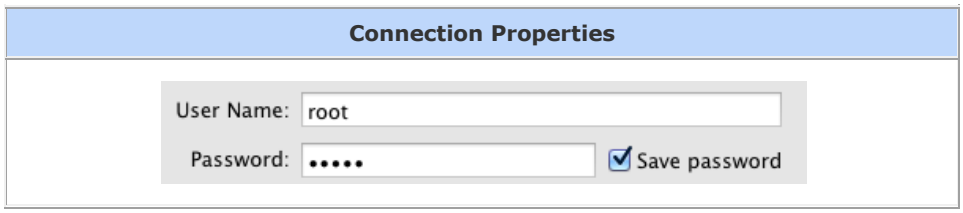
To extract SQL from a backup file, open a database and select an existing backup file. Control-click and select **Extract SQL** from the popup menu.

Batch Job/Schedule (Available only in Full Version)

Navicat allows you to create a batch job for setting schedule to execute at one or more regular intervals, beginning and ending at a specific date and time. Batch job can be created for Query, Backup, Data Transfer, Data Synchronization, Import and Export from databases. You can define a list of actions to be performed within one batch job, either run it manually or at the specified time/periodically. Click to open an object list for **Schedule**.

Click **Setup Schedule** to set schedule for batch job.

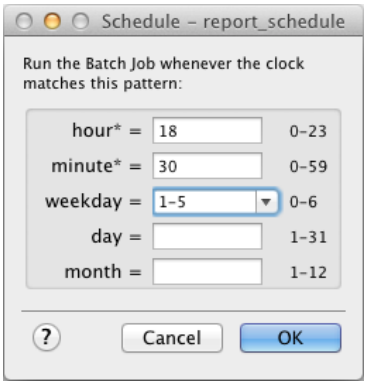
Note: Please save the batch job before setting schedule. Passwords must be saved in Connection Properties before running your schedule. User must log on the computer to run the schedule, even the computer is sleeping or switching user can run the schedule properly. Also, Navicat must be installed in /Applications or ~/Applications folder.



Setup Schedule

The pattern follows a crontab entry's format. If a field is left without a value, then all the values will be used. For example, if the "weekday" field is empty, then the system will treat the field to be entered with "0, 1, 2, 3, 4, 5, 6". Use commas to separate values. For example, "0, 1, 3, 6". Use hyphen, without spaces to indicate values. For example, "0-4".

Example: The batch job will be executed at 6:30pm every weekday.



Tips: What is crontab?

For commands that need to be executed repeatedly (e.g., hourly, daily, or weekly), you can use the crontab command. The crontab command creates a crontab file containing commands and instructions for the cron daemon to execute. You can use the crontab command with the following options:

crontab -a filename	Install filename as your crontab file. On many systems, this command is executed simply as crontab filename (i.e., without the -a option).
crontab -e	Edit your crontab file, or create one if it doesn't already exist.
crontab -l	Display your crontab file.
crontab -r	Remove your crontab file.
crontab -v	Display the last time you edited your crontab file. (This option is only available on a few systems.)
crontab -u user	Used in conjunction with other options, this option allows you to modify or view the crontab file of user. When available, only administrators can use this option.

Each entry in a crontab file consists of six fields, specifying in the following order:

minute(s) hour(s) day(s) month(s) weekday(s) command(s)

The fields are separated by spaces or tabs. The first five are integer patterns and the sixth is the command to execute.

Field	Value	Description
minute	0-59	The exact minute that the command sequence executes
hour	0-23	The hour of the day that the command sequence executes
day	1-31	The day of the month that the command sequence executes
month	1-12	The month of the year that the command sequence executes
weekday	0-6	The day of the week that the command sequence executes (Sunday = 0, Monday = 1, Tuesday = 2, and so forth)
command	Special	The complete sequence of commands to execute. The command string must conform to Bourne shell syntax. Commands, executables (such as scripts), or combinations are acceptable.

General Settings for Batch Job/Schedule

Move objects from the **Available Jobs** list to the **Selected Jobs** list by using ✚button or double-clicking them. To delete the objects from the selected jobs list, remove them in the same way. You are allowed to run profiles from different servers in a single batch job/schedule.

To rearrange the sequence of the selected jobs, drag to the desired location.

To backup whole server, you can select the connection and choose **Backup Server xxx**. (To backup your connection settings, see Migrate Navicat to new computer.)

To find Data Transfer or Data Synchronization profile, choose **Navicat** at the top on the left panel.

Advanced Settings for Batch Job/Schedule

Send email

Navicat allows you to generate and send personalized e-mails with results returned from a schedule. The resultset(s) can be emailed to multiple recipients. Check this option and enter required information.

From

Specify e-mail address of sender. For example, someone@navicat.com.

To, **CC**

Specify e-mail addresses of each recipient, separating them with a comma or a semicolon (;).

Subject

Specify the email subject with customized format.

Body

Write email content.

Include Log

Check this option to include the messages indicating the running process.

Host (SMTP Server)

Enter your Simple Mail Transfer Protocol (SMTP) server for outgoing messages.

Port

Enter the port number you connect to your outgoing e-mail (SMTP) server. Default value is **25**.

Use Authentication

Check this option and enter **User Name** and **Password** if your SMTP server requires authorization to send email.

Secure Connect

Specify the connection to use **TLS**, **SSL** secure connection or none.

With Attachment(s)

Exported file can be added to the batch job as mail attachment. Check this option to attach it.

Send Test Mail

Navicat will send you a test mail indicating success or failure.

Batch Job Converter (Available only in Navicat Premium)

Navicat Premium allows you to convert saved batch jobs from either Navicat for MySQL, Navicat for Oracle, Navicat for PostgreSQL, Navicat for SQLite, Navicat for SQL Server and Navicat for MariaDB to it. Control-click in the schedule object list and select **Batch Job Converter** from the popup menu.

Create schedules

Check this option to copy the existing schedule of selected batch jobs in Navicat to Navicat Premium.

Delete jobs from original application

Check this option if you want to delete the original batch jobs in Navicat. If the original batch job is deleted, the scheduled batch job will not work until it is set again in Navicat Premium or the original application.

Overwrite existing jobs

Check this option if you want to overwrite the existing batch jobs in Navicat Premium.

Click **Start** button to start the import process.

Dump SQL File/Execute SQL File

Navicat allows you to backup and restore your database/schema/table(s) using the **Dump SQL File** and **Execute SQL File** features.

To backup your database/schema/table(s), control-click it and select **Dump SQL File** -> **Structure + Data** or **Structure Only** from the popup menu.

To restore your database/schema/table(s) or execute SQL file, control-click the connection/database/schema and select **Execute SQL File** from the popup menu.

Hint: You can drag a .sql file to the a database/schema in the connection tree. Navicat will popup the **Execute SQL File** window.

View Database/Schema/Table Structure (Available only in Full Version)

Navicat allows you to view and print database, schema and table structures. Control-click the database/schema/table(s) and select **View Table Structure** from the popup menu. If you select database/schema, you can choose **In HTML Format**.

Console

Console allows you to use a command-line interface. In other words, it provides interactive text-based screen for you to query input and result output from database. Open the connection and select **Tools** -> **Console** from the main menu or press CMD-SHIFT-C.

Hint: You are allowed to open multiple console windows which each represents different connection.

Note: For Oracle server, you have to have **SQL*Plus** executable in order to get this works. By default, Navicat will look for the SQL*Plus under client folder (e.g. ORACLE_HOME/bin). . However, if Navicat cannot locate SQL*Plus under the SQL*Plus Default Path, you are prompted to locate the executable.
SQL*Plus does not support Unicode.

Server Security

Navicat provides security management tool for MySQL, Oracle, PostgreSQL, SQL Server and MariaDB. You can add, edit, delete users, grant/revoke privileges on the selected database and their database objects. Click to open an object list for **User**. The user object list displays all users that exist in the server.

MySQL/MariaDB Security

Information about user privileges is stored in the **user**, **db**, **host**, **tables_priv**, **columns_priv**, and **procs_priv** tables in the **mysql** database (that is, in the database named mysql). The MySQL server reads the contents of these tables when it starts.

MySQL access control involves two stages when you run a client program that connects to the server:

Stage 1: The server checks whether it should allow you to connect.

Stage 2: Assuming that you can connect, the server checks each statement you issue to determine whether you have sufficient privileges to perform it. For examples: Create table privilege, Drop table privilege or Alter table privilege.

The server uses the **user**, **db**, and **host** tables in the **mysql** database at both stages of access control.

MySQL/MariaDB User Designer

General Properties

User name

Set name for user.

Host

A host name where the database is situated or the IP address of the server.

Password

Set **Password** and re-type it in the **Confirm Password** text box.

Advanced Properties

Maximum Queries Per Hour, **Maximum Updates Per Hour**, **Maximum Connections Per Hour**

These options limit the number of queries, updates, and logins a user can perform during any given one-hour period. If they are set as 0 (the default), this means that there is no limitation for that user.

Maximum User Connection

This option limits the maximum number of simultaneous connections that the account can make. If it is set as 0 (the default), the *max_user_connections* system variable determines the number of simultaneous connections for the account.

Use OLD_PASSWORD encryption

The password hashing mechanism was updated in MySQL 4.1 to provide better security and to reduce the risk of passwords being

intercepted. However, this new mechanism is understood only by MySQL 4.1 (and newer) servers and clients, which can result in some compatibility problems. A 4.1 or newer client can connect to a pre-4.1 server, because the client understands both the old and new password hashing mechanisms. However, a pre-4.1 client that attempts to connect to a 4.1 or newer server may run into difficulties.

Enable this option if you wish to maintain backward compatibility with pre-4.1 clients under circumstances where the server would otherwise generate long password hashes. The option does not affect authentication (4.1 and later clients can still use accounts that have long password hashes), but it does prevent creation of a long password hash in the *user* table as the result of a password-changing operation.

SSL Type

MySQL can check X509 certificate attributes in addition to the usual authentication that is based on the username and password. To specify SSL-related options for a MySQL account, use the *REQUIRE* clause of the *GRANT* statement.

ANY	This option tells the server to allow only SSL-encrypted connections for the account.
X509	This means that the client must have a valid certificate but that the exact certificate, issuer, and subject do not matter. The only requirement is that it should be possible to verify its signature with one of the CA certificates.
SPECIFIED	**Issuer** This places the restriction on connection attempts that the client must present a valid X509 certificate issued by CA *issuer*. If the client presents a certificate that is valid but has a different issuer, the server rejects the connection. Use of X509 certificates always implies encryption, so the SSL option is unnecessary in this case. **Subject** This places the restriction on connection attempts that the client must present a valid X509 certificate containing the subject *subject*. If the client presents a certificate that is valid but has a different subject, the server rejects the connection. **Cipher** This is needed to ensure that ciphers and key lengths of sufficient strength are used. SSL itself can be weak if old algorithms using short encryption keys are used. Using this option, you can ask that a specific cipher method is used to allow a connection.

Server Privileges

In the grid, check **Grant** option against the server privilege listed in **Privilege** to assign this user to have that privilege. Multiple privileges can be granted.

Object Privileges

To edit specific object privileges for user, click ✚to open the window and follow the steps below:

1. Expand the node in the tree view until reaching to the target object.

2. Check the object to show the grid on the right panel.
3. In the grid, check **Grant** option against the privilege listed in **Privilege** to assign this user to have that privilege. Multiple privileges can be granted.

Oracle Security

Oracle manages database access permissions using users and roles. Users own schema objects (for example, tables, views) and can assign privileges on those objects to other users to control who has access to which objects.

In addition to the user accounts that you create, the database includes a number of user accounts that are automatically created upon installation. Administrative accounts: **SYS**, **SYSTEM**, **SYSMAN**, and **DBSNMP**. Administrative accounts are highly privileged accounts to perform administrative tasks such as starting and stopping the database, managing database memory and storage, creating and managing database users, and so on. Your database may also include sample schemas (**SCOTT**, **HR**, **OE**, **OC**, **PM**, **IX** and **SH**), which are a set of interlinked schemas that enable Oracle documentation and Oracle instructional materials to illustrate common database tasks.

When you create a database object, you become its owner. By default, only the owner of an object can do anything with the object. In order to allow other users to use it, privileges must be granted. (However, users that have the superuser attribute can always access any object.)

Ordinarily, only the object's owner (or a superuser) can grant or revoke privileges on an object. However, it is possible to grant a privilege **Admin Option**/**Grant Option**, which gives the recipient the right to grant it in turn to others. If the grant option is subsequently revoked then all who received the privilege from that recipient (directly or through a chain of grants) will lose the privilege.

Note: The special name **PUBLIC** is accessible to every database user, all privileges and roles granted to **PUBLIC** are accessible to every database user.

Oracle User Designer

General Properties

User name

Set name for user.

Authentication

Select the authentication method.

Password	**Password** Set **Password** and re-type it in the **Confirm Password** text box. **Expired Password** Expire the user's password. This setting forces the user or the DBA to change the password before the user can log in to the database.

External	An external user must be authenticated by an external service, such as an operating system or a third-party service.
Global	A global user must be authorized by the enterprise directory service (Oracle Internet Directory). **X.500 Distinguished Name** Enter the X.509 name at the enterprise directory service that identifies this user.

Default Tablespace

Choose the default tablespace for objects that the user creates.

Temporary Tablespace

Choose the tablespace or tablespace group for the user's temporary segments.

Profile

Choose the profile that assign to the user.

Locked account

Lock the user's account and disable access.

Roles

In the grid, check **Granted**, **Admin Option** or **Default** option against the role listed in **Role Name** to assign this user to be a member of selected role. Multiple roles can be granted.

Quotas

In the grid, specify the maximum amount of space the user can allocate in the tablespaces. Enter the **Quota** and choose the **Unit** of the **Tablespace**. **Unlimited** lets the user allocate space in the tablespace without bound. Multiple tablespaces can be set.

System Privileges

In the grid, check **Grant** or **Admin Option** option against the server privilege listed in **Privilege** to assign this user to have that privilege. Multiple privileges can be granted.

Object Privileges

To edit specific object privileges for user, click ✚to open the window and follow the steps below:

1. Expand the node in the tree view until reaching to the target object.
2. Check the object to show the grid on the right panel.
3. In the grid, check **Grant** or **Grant Option** option against the privilege listed in **Privilege** to assign this user to have that privilege. Multiple privileges can be granted.

Oracle Role Designer

General Properties

Role name

Set name for role.

Authentication

Select the authentication method.

Password	**Password** Set **Password** and re-type it in the **Confirm Password** text box.
External	An external user must be authenticated by an external service, such as an operating system or a third-party service, before enabling the role.
Global	A global user must be authorized to use the role by the enterprise directory service before the role is enabled at login.
Not Identified	The role is authorized by the database and that no password is required to enable the role.

Roles

In the grid, check **Granted** or **Admin Option** option against the role listed in **Role Name** to assign this role to be a member of selected role. Multiple roles can be granted.

Members

In the grid, check **Granted** or **Admin Option** option against user listed in **Name** to assign the selected user to be a member of this role. Multiple users can be granted.

System Privileges

In the grid, check **Grant** or **Admin Option** option against the server privilege listed in **Privilege** to assign this role to have that privilege. Multiple privileges can be granted.

Object Privileges

To edit specific object privileges for role, click ✚to open the window and follow the steps below:

1. Expand the node in the tree view until reaching to the target object.
2. Check the object to show the grid on the right panel.
3. In the grid, check **Grant** or **Grant Option** option against the privilege listed in **Privilege** to assign this role to have that privilege. Multiple privileges can be granted.

PostgreSQL Security

PostgreSQL manages database access permissions using users and groups. Users own database objects (for example, tables) and can assign privileges on those objects to other users to control who has access to which objects.

Note: Starting from PostgreSQL version 8.1, users and groups were no longer distinct kinds of entities, now there are only roles. Any role can act as a user, a group, or both. The concept of roles subsumes the concepts of users and groups.

Only a superuser (a user who is allowed all rights) can add/delete users. PostgreSQL installs a single superuser by default named **postgres**. All other users must be added by this user, or by another subsequently added superuser.

When you create a database object, you become its owner. By default, only the owner of an object can do anything with the object. In order to allow other users to use it, privileges must be granted. (However, users that have the superuser attribute can always access any object.)

Ordinarily, only the object's owner (or a superuser) can grant or revoke privileges on an object. However, it is possible to grant a privilege **With Grant Option**, which gives the recipient the right to grant it in turn to others. If the grant option is subsequently revoked then all who received the privilege from that recipient (directly or through a chain of grants) will lose the privilege.

Note: The special name **public** can be used to grant a privilege to every role (user/group) on the system.

PostgreSQL Server 7.3 to 8.0

PostgreSQL version 7.3 to 8.0 manages database access permissions using **Users** and **Groups**.

PostgreSQL User Designer

General Properties

User Name

Set name for user.

User ID

Specify an ID for the user. This is normally not necessary, but may be useful if you need to recreate the owner of an orphaned object. If this is not specified, the highest assigned user ID plus one (with a minimum of 100) will be used as default

Password

Set **Password** and re-type it in the **Confirm Password** text box.

Note: If you do not plan to use password authentication you can omit this option, but then the user will not be able to connect if you decide to switch to password authentication.

Password Encryption

This option control whether the password is stored **ENCRYPTED** or **UNENCRYPTED** in the system catalogs. (If neither is specified, the default behavior is determined by the configuration parameter *password_encryption*.)

Expiry Date

Set a date and time after which the user's password is no longer valid. If this clause is omitted the password will be valid for all time.

Can create databases

Check this option to define the user to be allowed to create databases.

Superuser

Check this option to define the user as a superuser.

Members of

In the grid, check **Granted** option against the group listed in **Group Name** to assign this user to be a member of selected group. Multiple groups can be granted.

Object Privileges

To edit specific object privileges for user, click + to open the window and follow the steps below:

1. Expand the node in the tree view until reaching to the target object.
2. Check the object to show the grid on the right panel.
3. In the grid, check **Grant** or **Grant Option** option against the privilege listed in **Permission** to assign this user to have that privilege. Multiple privileges can be granted.

PostgreSQL Group Designer

General Properties

Group Name

Set name for group.

Group ID

Specify an ID for the group. This is normally not necessary, but may be useful if you need to recreate a group referenced in the permissions of some object. If this is not specified, the highest assigned group ID plus one (with a minimum of 100) will be used as default.

Users

In the grid, check **Granted** option against the user listed in **User Name** to assign selected user to be a member of this group. Multiple users can be granted.

Object Privileges

To edit specific object privileges for group, click + to open the window and follow the steps below:

1. Expand the node in the tree view until reaching to the target object.
2. Check the object to show the grid on the right panel.
3. In the grid, check **Grant** option against the privilege listed in **Permission** to assign this group to have that privilege. Multiple privileges can be granted.

PostgreSQL Server 8.1 to 9.2

Starting from PostgreSQL version 8.1, users and groups were no longer distinct kinds of entities, now there are only **Roles**. Any role can act as a user, a group, or both. The concept of roles subsumes the concepts of users and groups.

PostgreSQL Role Designer

General Properties

Role Name

Set name for role.

Role ID

Specify an ID for the role. This is normally not necessary, but may be useful if you need to recreate the owner of an orphaned object. If this is not specified, the highest assigned role ID plus one (with a minimum of 100) will be used as default.

Note: In PostgreSQL versions 8.1 or above, the specified ID will be ignored, but is accepted for backwards compatibility.

Can login

Check this option to create a role that allow to login. A role having this option can be thought of as a user. Roles without this attribute are useful for managing database privileges, but are not users in the usual sense of the word.

Password

Set **Password** and re-type it in the **Confirm Password** text box.

Note: If you do not plan to use password authentication you can omit this option, but then the role will not be able to connect if you decide to switch to password authentication.

Password Encryption

This option control whether the password is stored **ENCRYPTED** or **UNENCRYPTED** in the system catalogs. (If neither is specified, the default behavior is determined by the configuration parameter *password_encryption*.)

Connection Limit

If role can log in, this specifies how many concurrent connections the role can make. -1 (the default) means no limit.

Expiry Date

Set a date and time after which the role's password is no longer valid. If this clause is omitted the password will be valid for all time.

Can create databases

Check this option to define a role's ability to create databases.

Superuser

Check this option to determine the new role is a superuser, who can override all access restrictions within the database.

Can modify catalog directly

Check this option to allow a role's ability to update system catalog.

Inherit rights from parent roles

Check this option to determine whether a role inherits the privileges of roles it is a member of.

Can create roles

Check this option to allow creating roles.

Members of

In the grid, check **Granted** or **Admin Option** option against the role listed in **Role Name** to assign this role to be a member of selected role. Multiple roles can be granted.

Members

In the grid, check **Granted** or **Admin Option** option against the role listed in **Role Name** to assign the selected role to be a member of this role. Multiple roles can be granted.

Object Privileges

To edit specific object privileges for role, click ✚to open the window and follow the steps below:

1. Expand the node in the tree view until reaching to the target object.
2. Check the object to show the grid on the right panel.
3. In the grid, check **Grant** or **Grant Option** option against the privilege listed in **Permission** to assign this role to have that privilege. Multiple privileges can be granted.

SQL Server Security

The SQL Server **sa** log in is a server-level principal. By default, it is created when an instance is installed. In SQL Server 2005 or above, the default database of **sa** is **master**. This is a change of behavior from earlier versions of SQL Server.

By default, the database includes a **guest** user when a database is created. Permissions granted to the **guest** user are inherited by users who do not have a user account in the database. The **guest** user cannot be dropped, but it can be disabled by revoking its CONNECT permission. The CONNECT permission can be revoked by executing *REVOKE CONNECT FROM GUEST* within any database other than **master** or **tempdb**.

In SQL Server, the concept for permissions is using principals and securables. Principals are the individuals, groups, and processes granted access to SQL Server. Securables are the server, database, and objects the database contains. Principals can be arranged in a hierarchy. To easily manage the permissions in your databases, SQL Server provides several roles which are security principals that group other principals. Database-level roles are database-wide in their permissions scope.

Windows-level principals

- Windows Domain Login
- Windows Local Login

SQL Server-level principal

- SQL Server Login

Database-level principals

- Database User
- Database Role
- Application Role

Login

SQL Server uses two ways to validate connections to SQL Server databases: Windows Authentication and SQL Server Authentication. SQL Server Authentication uses login records to validate the connection. A Login object exposes a SQL Server login record.

Server Role

Server-level roles are also named fixed server roles because you cannot create new server-level roles and the permissions of fixed server roles cannot be changed. You can add SQL Server logins, Windows accounts, and Windows groups into server-level roles. Each member of a fixed server role can add other logins to that same role.

Database User

To gain access to a database, a login must be identified as a database user. The database user is usually known by the same name as the login, but you can create a database user (for a login) with a different name.

Database Role

Fixed database roles are defined at the database level and exist in each database. You can add any database account and other SQL Server roles into database-level roles. Each member of a fixed database role can add other logins to that same role.

Application Role

An application role is a database principal that enables an application to run with its own, user-like permissions. You can use application roles to enable access to specific data to only those users who connect through a particular application. Unlike database roles, application roles contain no members and are inactive by default.

SQL Server Login Designer

General Properties for SQL Server 2000

Login Name

Set name for login.

Authentication Type

Select the authentication type.

SQL Server Authentication	**Password** Set **Password** and re-type it in the **Confirm Password** text box. **Specify Old Password** Check this option to enter the old password used by this account. **Default Database** Select the default database when login. **Default Language** Select the default display language when login.
Windows Authentication	**Default Database** Select the default database when login. **Default Language** Select the default display language when login.

General Properties for SQL Server 2005 or later

Login Name

Set name for login.

Authentication Type

Select the authentication type.

SQL Server Authentication	**Password** Set **Password** and re-type it in the **Confirm Password** text box. **Specify Old Password** Check this option to enter the old password used by this account. **Enforce password policy** You can check this option to force password to follow password policy of SQL Server. **Enforce password expiration** You can check this option to force password to have expiry date. **User must change password at next login** You can check this option to force user to change password everytime when login. **Default Database** Select the default database when login. **Default Language**

	Select the default display language when login. **Locked** Uncheck this option to unlock the login. Note: Support from SQL Server 2008 or later.
Windows Authentication	**Default Database** Select the default database when login. **Default Language** Select the default display language when login.
Mapped to Certificate	**Certificate Name** Select the certificate name.
Mapped to Asymmetric Key	**Asymmetric Key Name** Select the asymmetric key name.

Note: SQL Server contains features that enable you to create and manage certificates and keys for use with the server and database. You can use externally generated certificates or SQL Server can generate certificates. Certificates and asymmetric keys are both ways to use asymmetric encryption. There is no difference between the two mechanisms for the cryptographic algorithm, and no difference in strength given the same key length.

Credential

You can add credential on specific role for this login. A credential is a record that contains the authentication information (credentials) required to connect to a resource outside SQL Server. This information is used internally by SQL Server.

Enabled

Check to enable the login.

General Properties for SQL Azure

Login Name

Set name for login.

Password

Set **Password** and re-type it in the **Confirm Password** text box.

Specify Old Password

Check this option to enter the old password used by this account.

Enabled

Check to enable the login.

Roles

In the grid, check the server role to assign this server login to be a member of selected server role. Multiple roles can be granted.

Note: Every SQL Server login belongs to the **public** server role. When a server principal has not been granted or denied specific permissions on a securable object, the user inherits the permissions granted to **public** on that object. Only assign **public** permissions on any object when you want the object to be available to all users.
SQL Azure does not support.

User Mapping

In the Grid, check the **Database** and enter the **User** and **Default Schema** to create user for login the database and specify the first schema will be searched by the server.

Server Permissions

You can check **Grant**, **With Grant Option** or **Deny** against the server permissions listed in **Permission** to assign this login to have that permission. Multiple permissions can be granted.

Note: Support from SQL Server 2005 or later.

Endpoint Permissions

You can check **Alter**, **Connect**, **Control**, **Take Ownership** or **View Definition** against the endpoint listed in **Endpoint** to assign this login to have that endpoint permission. Multiple permissions can be granted.

Note: Support from SQL Server 2005 or later.

Login Permissions

You can check **Alter**, **Control**, **Impersonate** or **View Definition** against the server login listed in **Login** to assign this server login to have that login permission. Multiple permissions can be granted.

Note: Support from SQL Server 2005 or later.

SQL Server Server Role Designer

Members

In the grid, check **Member** against the server role listed in **Name** to assign the selected server role to be a member of this server role. Multiple roles can be granted.

Note: SQL Azure does not support server role.

SQL Server Database User Designer

General Properties for SQL Server 2000

User Name

Set name for database user.

Login Name

Assign SQL Server login that this database user uses. When this SQL Server login enters the database, it will retrieve the information of this database user.

General Properties for SQL Server 2005 or later

User Name

Set name for database user.

Authentication Type

Select the type for database user.

Login	**Login Name** Assign SQL Server login that this database user uses. When this SQL Server login enters the database, it will retrieve the information of this database user. **Default Schema** You can specify the first schema that will be searched by the server for this database user.
Certificate	**Certificate Name** Specify the certificate for this database user.
Asymmetric Key	**Asymmetric Key Name** Specify the asymmetric key for this database user.
Without Login	**Default Schema** You can specify the first schema that will be searched by the server for this database user.

General Properties for SQL Azure

User Name

Set name for database user.

Authentication Type

Select the type for database user.

Login	**Login Name** Assign SQL Server login that this database user uses. When this SQL Server login enters the database, it will retrieve the information of this database user.
Without Login	Specify this database user not be mapped to an existing login.

Roles

In the grid, check **Granted** against the database role listed in **Role Name** to assign this database user to be a member of selected database role. Multiple roles can be granted.

Every database user belongs to the **public** database role. When a user has not been granted or denied specific permissions on a securable, the user inherits the permissions granted to **public** on that securable.

Database Permissions

In the grid, check **Grant**, **Grant Option** or **Deny** against the database permission listed in **Permission** to assign this database user to have that permission on the database. Multiple permissions can be granted.

Note: SQL Server 2000 does not have Grant Option column.

Object Permissions

To edit specific object permission for database user, click ✚to open the window and follow the steps below:

1. Expand the node in the tree view until reaching to the target object.
2. Check the object to show the grid on the right panel.
3. In the grid, check **Grant**, **Grant Option** or **Deny** against the permission listed in **Permission** to assign this database user to have that permission. Multiple permissions can be granted.

Note: SQL Azure does not support the **Comment** tab.

SQL Server Database Role Designer

General Properties

Role name

Set name for database role.

Owner

You can enter the owner for this database role. This owner can be database user or database role. If the owner is not specify, this database role will be owned by the user who executes the *CREATE ROLE*.

Members

In the grid, check **Member** against the database user/role listed in **Name** to assign the selected database user/role to be a member of this database role. Multiple roles can be granted.

Member Of

In the grid, check **Member of** against the database role/application role listed in **Name** to assign this database role to be a member of the selected database role/application role. Multiple roles can be granted.

Database Permissions

In the grid, check **Grant**, **Grant Option** or **Deny** against the permission listed in **Permission** to assign this database role to have that permission. Multiple permissions can be granted.

Note: SQL Server 2000 does not have Grant Option column.

Object Permissions

To edit specific object permission for database role, click ✚to open the window and follow the steps below:

1. Expand the node in the tree view until reaching to the target object.
2. Check the object to show the grid on the right panel.
3. In the grid, check **Grant**, **Grant Option** or **Deny** against the permission listed in **Permission** to assign this database role to have that permission. Multiple permissions can be granted.

Note: SQL Server 2000 and SQL Azure does not support the **Comment** tab.

SQL Server Application Role Designer

General Properties

Role name

Set name for role.

Password

Set **Password** and re-type it in the **Confirm Password** text box.

Default Schema

You can specify the first schema that will be searched by the server for this application role.

Note: Support from SQL Server 2005 or later.

Database Permissions

In the grid, check **Grant**, **Grant Option** or **Deny** against the permission listed in **Permission** to assign this application role to have that permission. Multiple permissions can be granted.

Note: SQL Server 2000 does not have Grant Option column.

Object Permissions

To edit specific object permission for application role, click ✚to open the window and follow the steps below:

1. Expand the node in the tree view until reaching to the target object.
2. Check the object to show the grid on the right panel.
3. In the grid, check **Grant**, **Grant Option** or **Deny** against the permission listed in **Permission** to assign this application role to have that permission. Multiple permissions can be granted.

Note: SQL Server 2000 does not support the **Comment** tab.
SQL Azure does not support application role.

Privilege Manager

Besides setting privileges in each user, the **Privilege Manager** provides another view on privileges in connection and its database objects.

Control-click the connection and select **Set Privileges** from the popup menu and follow the steps below:

1. Expand the node in the tree view until reaching to the target object.
2. Choose the object and click ✚to open the window.
3. Check the user to show the grid on the right panel.
4. In the grid, check the relevant privilege against the privilege listed in **Privilege** to assign the selected user to have that object privilege. Multiple privileges can be granted.

Useful Tools

Navicat provides variety of tools that improve user experience when using Navicat, which are **Object Information**, **Connection Colorings**, **Search Filter** and more.

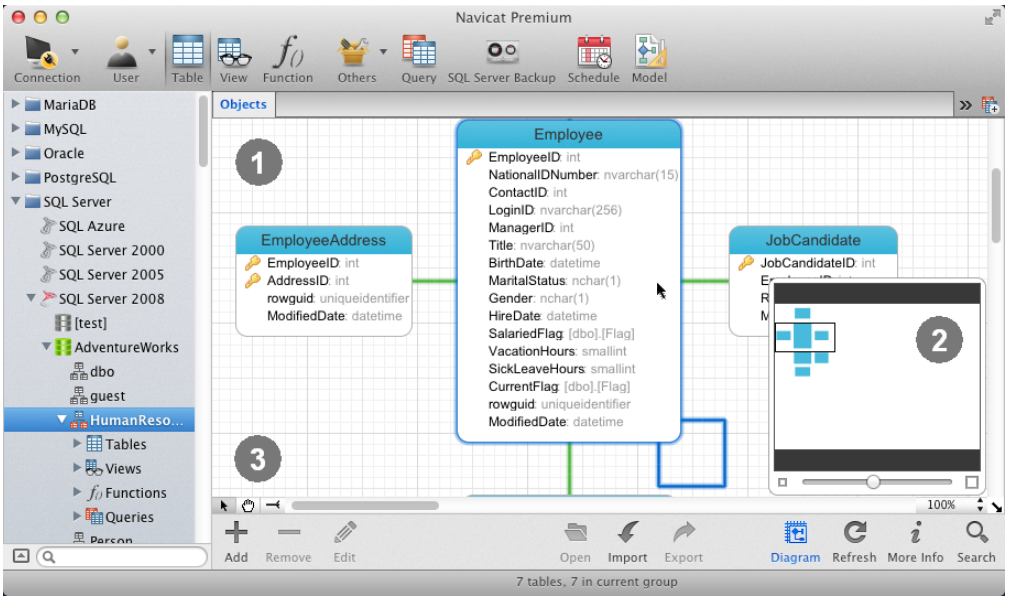
ER Diagram (Available only in Full Version)

Besides **List** and **Detail** views, Navicat enhances table viewing to a new **ER Diagram** view. In this ER Diagram view, you can view table fields and relationships between tables in a database and/or schema graphically. It also allows adding foreign key constraints to tables directly.

Note: Only tables provide ER Diagram view in Navicat. Other database objects only provide List view and Detail view.

Select **View** -> **ER Diagram** from the main menu or click **Diagram** from the table object list toolbar. An ER diagram will be created automatically if the selected database contains tables.

Hint: ER Diagram files are stored under Settings Location.



Object List

Display table fields and relationships between tables in a database/schema.

Note: Double-click a table in ER Diagram view will open the Table Designer, while double-click a table in List view and Detail view will open the Table Viewer. The tabs and options in the designer depend on the diagram database type you have chosen. For the settings of different tabs, see Server Objects.

To set color to selected tables, control-click the table and select **Color** from the popup menu.

To add a relation, click from the bottom toolbar. Drag the source table field and drop to the target table field.

To edit a relation, control-click a relation and select **Design Relation** from the popup menu.

To delete a relation, control-click a relation and select **Delete Relation** from the popup menu.

Select relation in diagram, then press and hold the SHIFT key. Click on the relation to add vertex to relation.

Select relation in diagram, then press and hold the SHIFT key. Click on the vertex to delete vertex on relation.

Auto Layout

To regenerate the ER Diagram. Control-click the object list and select **Auto Layout** from the popup menu.

Paper Dimension

Control-click the object list and select **Page Dimension** from the popup menu. Corresponding paper dimension will reflect in Navigator.

2 Navigator

To zoom in or zoom out the selected area of the diagram, adjust the slider of the Navigator. Same effect can be achieved with keyboard shortcuts:

Zoom In: [CMD-+] or [CMD-Mousewheel up]

Zoom Out: [CMD--] or [CMD-Mousewheel down]

3 Toolbar

Move Diagram

Click to switch to hand mode. Press and hold the SPACE key, then move the diagram.

New Relation

Click to create a relation between two table fields.

Object Information

In the object list, you can also view information of selected object. Select **View** -> **Object Information** from the main menu or press CMD-I or click **More Info** from the object list toolbar.

Note: The tabs depend on the object type you have chosen.

Information

Show the object information.

Columns

Shows columns in the table/view.

Using

Show the objects that the current object used.

Used by

Show the current object used by whom.

DDL

Show the DDL statement of the object.

Objects

Show the objects in the tablespace.

Preview

Show the sql statement in the query.

Member of

Show the roles that the user or the role assigned to.

Members

Show the members of the role.

Server Monitor (Available only in Full Version)

Navicat provides **Server Monitor** to view properties of selected server(s). Select **Tools** -> **Server Monitor** and select the preferred server type from the main menu.

Note: Available only for MySQL, Oracle, PostgreSQL, SQL Server and MariaDB. SQL Azure does not support.

Process List

Display a list of processes from all servers selected.

To stop the selected process, click **End Process** button or press CMD-E.

Auto refresh

If you want to take action on auto-refreshing the server in assigned seconds, choose **Edit** -> **Set Refresh Rate** and enter an auto refresh value. To disable auto refresh feature, choose **File** -> **Stop Auto Refresh** or press SHIFT-CMD-R.

Note: Effect will take once you assign the value.

The process list provides the following information depends on the database type you are chosen: MySQL, Oracle, PostgreSQL, SQL Server and MariaDB.

- Server name that is given while setting the connection.
- Process ID on the server.
- Serial number of the process.
- Current user who log in to the server.
- Host from which the user is connected.
- Database that the user is currently used.
- Last command that was issued by the user.
- Time, state and info of the process.

- CPU Time and state of the process.

Variables

Display the list of all server variables and their values.

Note: Available only for MySQL, Oracle, PostgreSQL and MariaDB.

Hint: To edit variable value in MySQL and Oracle servers, click to open editor for editing. The value in PostgreSQL server cannot be edited here. (Those variables can be set using the *SET* statement, by editing the *postgresql.conf* configuration file.)

Status

Display the list of all server status and their values.

Note: Available only for MySQL, Oracle, PostgreSQL and MariaDB.

Virtual Grouping (Available only in Full Version)

Virtual Group aims to provide a platform for logical grouping objects by categories, so that all objects are effectively preserved.

Virtual Grouping can be applied on Connection, Table, View, Function, Query, Backup, Schedule and Model.

Control-click in the connection tree/object list and select **Manage Group** -> **New Group** from the popup menu to create a new group.

To move object(s) to a group, control-click the object(s) and select **Manage Group** -> **Add to Group** from the popup menu or drag and drop the object(s) into the group.

To move object(s) back to the top-level, control-click the object(s) and select **Manage Group** -> **Remove From Group** from the popup menu or drag and drop the object(s) to target level in the connection tree.

Select **View** -> **Flatten Connection**/**Flatten Object List** from the main menu to hide connection/object groups.

Connection Colorings

Navicat provides highlighting connections by colors for identifying connections and their database objects. The highlighted color displays in the connection tree and next to the database object name in the tab bar.

To highlight a connection, control-click the connection and select **Color** from the popup menu.

Favorites (Available only in Full Version)

Favorites are links to database objects that you visit frequently. By adding a path to your favorites list, you can go to that database objects with a single click, instead of having to navigate the connection and database and/or schema in the connection tree.

To add a link to the favorites list, open a database object and choose **Favorites** -> **#. <Empty>** -> **Add To Favorite #** or press OPTION-SHIFT-#. Enter **Favorite Name** and select **Favorite ID**.

To open a database object from the favorites list, select **Favorites** -> **favorite_name** from the main menu or press CTRL-SHIFT-#.

Select **Favorites** -> **favorite_name** -> **Clear Favorite #** from the main menu to remove a link from the favorites list.

Select **Favorites** -> **Clear Favorites** -> from the main menu to remove all links from the favorites list.

Note: # represents 0, 1, 2, 3, 4, 5, 6, 7, 8 or 9.

Find in Database/Schema (Available only in Full Version)

Navicat provides a **Find in Database/Schema** feature offers searching table and view records within a database and/or schema. Control-click the database/schema and select **Find in Database/Find in Schema** from the popup menu.

Enter keyword and select the search criteria. Click **Find** button and then double-click table/view in the list to view the record.

Search Filter

Navicat provides filters for your objects in the object list and connections in the connection tree.

Object Filter allows you to filter objects in the object list or model that names contain the filter string.

Click **Search** in the Navicat main window or click in Model Designer and specify a filter string.

Click to have additional options for the filter.

Ignore Case

Disable case sensitive filter.

Contains

Return the objects that contain the filter string.

Start With

Return the objects that start with the filter string.

Whole Words

Return the objects that match the entire word of the filter string.

Ends With

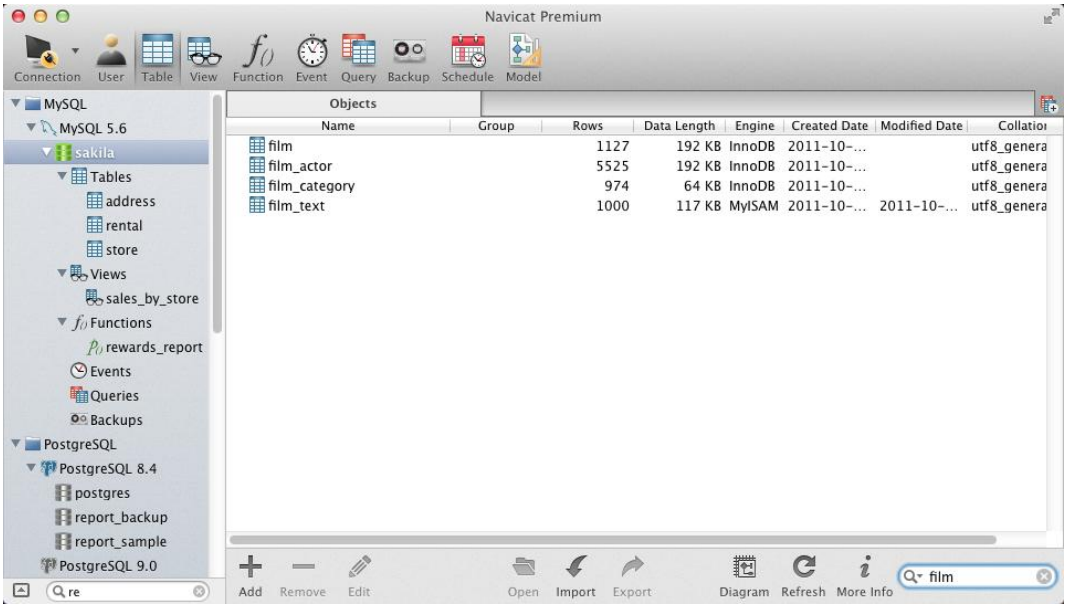
Return the objects that end with the filter string.

To remove the filter, simply delete the filter string.

Tree Filter allows you to filter connection in the connection tree that names contain the filter string.

Specify a filter string in the text box in the connection tree. If connection has opened, the filter will also apply to its database objects.

To remove the filter, simply delete the filter string.



Preferences

Navicat provides a complete user interface customization with various options for all tools. Select **Navicat Premium** -> **Preferences** from the main menu.

General

Allow opening multiple instances

With this option is on, you allow opening multiple instances of the same selected window.

Show table hint

While you roll the mouse pointer over a table within the object list, you could get a popup hint giving details about the table structure.

Show tables in connections tree

Display database objects using the tree structure in connection tree. To expand node, click ▶ or double-click the node.

Hint: Reopen the database/schema to take effect.

Click to refresh object list

Refresh the object list whenever you click on the objects.

Show system items (PostgreSQL, SQL Server)

Check this option to show all the system objects such as *information_schema* and *pg_catalog* schemas.

Hint: Reopen the database/schema to take effect.

Show auto index (SQLite)

Check this option to show auto index generated for SQLite table in the index object list.

Ask to save new queries/profiles before closing

With this option is on, Navicat will prompt you to save new queries or profiles every time when you quit the relevant sub-window.

Automatically check for updates

Check this option to allow Navicat checks for new version automatically at a selected time.

Include anonymous system profile

Check this option to send us your system information, such as your Mac OS version to improve our Navicat when Navicat checks for updating.

Tabs

Default open in:

Open new windows to **Main window**, **Tabbed window** or **New window**.

Warn me when closing multiple tabs

A warning message will remind you while you are closing multiple tabs.

Select the newly created tab

The latest opened tab will get focus on the tabbed window.

Always show the tab bar

Show tab bar.

Open new tab in latest window

If you choose **Tabbed window** in **Default open in** option, it allows you to check this option to open new tab in latest window. Otherwise, new tab will be opened in main window even there is a new window opened for objects.

Grids

Limit records ☐ records per page

Check this option if you want to limit the number of records showed on each page in table grid globally. Otherwise, all records will be displayed in one single page. The number representing the number of records (e.g. 1000) showed per page in table grid.

Note: To adjust the settings for particular table, see Table Viewer.

Limit foreign key editor records ☐ records

Check this option if you want to limit the number of records showed on each page in foreign key data selection globally. Otherwise, all records will be displayed in one single page. The number representing the number of records (e.g. 100) showed per page in Foreign Key Data Selection.

Shading of Table Grid

Define the background layout of the data grid.

Default Row Height

Define the height of the row (e.g. 17) used in editor.

Note: To adjust the settings for particular table, see Formatting Table Grid.

Default Column Width

Define the width of the column (e.g. 100) used in editor.

Note: To adjust the settings for particular table, see Formatting Table Grid.

Show primary key warning

Check this option if you require notification while opening the table with no primary key being set.

Default show text

If this option is on, data which set as TEXT field type is visible in table grid. Otherwise, [TEXT] will shown.

Default show image

If this option is on, data which set as BLOB field type is visible in table grid. Otherwise, [BLOB] will shown

Trigger editing by double-click

With this option selected, double-clicking on a cell in table will start cell editing. While single-clicking, the row is highlighted.

Auto begin transaction

Check this option if you require auto commit of changing records in table grid. Otherwise, click **Commit** or **Rollback** buttons from the **Begin Transaction** button to commit or rollback the changes. See Table Viewer.

Fonts and Colors

Object List Font

Define the font and its size used by object list.

Data Grid Font

Define the font and its size used by grid in Table Viewer.

Editor Font

Define the font and its size used by editors.

Editor Color

This color settings allows you to format your SQL queries in SQL Editor with colored syntax highlighting for your SQL statements to improve readability.

Set font colors of the SQL Editor, uses to mark out different text fragments: Common, Keywords, Comments, Strings and Numbers. Just simply click on the color boxes and choose your desired color from the **Color-Selection** dialog.

File Paths

Write log for executed queries

Store all SQL statements of all the operations executed over databases and database objects in Navicat.

Hint: Restart Navicat to take effect.

Empty queries log file before Navicat starts

You can clear the log every time when Navicat launches.

Write log for batch jobs (Available only in Full Version)

Specify the location for a text file that stores information for Navicat command line process and all operations while running schedule.

Virtual Grouping File (Available only in Full Version)

Specify the location for a xml file that stores how the connections and objects are categorized in Navicat.

Hint: Restart Navicat to take effect.

SQL*Plus Default Path (Available only for Oracle)

Specify the location for SQL*Plus used in console of Oracle connection.

SQLite Dynamic Library Path

Specify the location for the SQLite Dynamic Library.

Hint: Restart Navicat to take effect.

Data Models Path (Available only in Full Version)

Specify the location for model files.

SQL Editors

Show line number

Display line numbers at the side of the editor for easily reference.

Use code folding

Code folding allows codes to collapse as a block and only the first line displayed in the editor, see Code Folding.

Use brace highlighting

Highlight the pair of braces when your cursor moves to either one brace for easily reference, see Brace Highlight.

Use syntax highlighting

Syntax highlight helps viewing codes clearly. Codes are highlighted in SQL Editor with different colors and fonts according to the categories they belong to.

Disable if file size larger than ☐ **MB**

The syntax highlighting feature can be limited by setting the maximum file size (e.g. 10) to increase performance.

Use code completion (Available only in Full Version)

When you type the . (dot) symbol between the object names, SQL Editor will offer you a popup list that showing some variants for the code completion, see Code Completion.

Delay

You can change the time the popup list takes to appear.

Use word completion (Available only in Full Version)

When you type the first character of words, SQL Editor will offer you a popup list that showing some variants for the word completion.

Delay

You can change the time the popup list takes to appear.

Autosave modified documents

Save automatically after modifications in SQL Editor by defining the interval (e.g. Every 30 seconds).

Tab Width

Enter the number of characters that a tab occupies, e.g. 5.

Models (Available only in Full Version)

Guess field types

With this option is on, Navicat will predict field types when you design fields without using Table Designer.

Highlight objects

With this option is on, when a mouse cursor hovers over an object, Navicat will highlight its border with blue color.

Highlight table with relations

With this option is on, when a mouse cursor hovers over a relation, Navicat will highlight it with blue or green color indicating relationships between tables.

Environments

Download/CD Version

ORACLE_HOME

Location of ORACLE_HOME for full client only. Instant client should leave it blank.

DYLD_LIBRARY_PATH

Location of the path which contains Oracle libraries for instant client and SQL*Plus (e.g. ORACLE_HOME/lib). Always required.

Use bundled instant client

Oracle Instant Client has already included in Navicat installation folder. Check this option to use the included one in Navicat installation folder, e.g. /Applications/Navicat Premium.app/Contents/OCI.

Oracle Instant Client is the simplest way to deploy a full Oracle Client application built with OCI, OCCI, JDBC-OCI, or ODBC drivers. It provides the necessary Oracle Client libraries in a small set of files. It provides the necessary Oracle Client libraries in a small set of files. You can also download **Oracle Instant Client** through -

Oracle Instant Client

http://www.oracle.com/technetwork/database/features/instant-client/index-097480.html

Download the appropriate Instant Client packages for your platform and the CPU. All installations REQUIRE the Basic or Basic Lite package. Unzip the packages and set the path points to it.

TNS_ADMIN

Location of the tnsnames.ora file (e.g. ORACLE_HOME/network/admin). It is optional. Required when using TNS connection.

Note: To change the path settings, you need to click the lock and enter your OS username and password to unlock the control. If you lock it back after changes, a dialog will be prompted to ask whether you want to save the settings. If you click **OK** after changes, it will just save without prompting any dialogs. If you wait until the lock timeout (300 seconds) after changes, your changes will be reverted.

Note: Available only for Oracle.

Hint: Restart Navicat to take effect.

App Store Version

This tab is removed from the App Store's version. To change the paths in App Store's version, please launch the application through **Terminal** and type the environment variables.

Navicat Version	Command
Navicat Premium	env TNS_ADMIN=~/Library/Containers/com.prect.NavicatPremium/Data/Library/Application\ Support/PremiumSoft\ CyberTech/ DYLD_LIBRARY_PATH=~/Library/Containers/com.prect.NavicatPremium/Data/Library/Application\ Support/PremiumSoft\ CyberTech/OCI arch -i386 /Applications/Navicat\ Premium.app/Contents/MacOS/Navicat\ Premium
Navicat Premium Essentials	env TNS_ADMIN=~/Library/Containers/com.prect.NavicatPremiumEssentials/Data/Library/Application\ Support/PremiumSoft\ CyberTech/ DYLD_LIBRARY_PATH=~/Library/Containers/com.prect.NavicatPremiumEssentials/Data/Library/Application\ Support/PremiumSoft\ CyberTech/OCI arch -i386 /Applications/Navicat\ Premium\ Essentials.app/Contents/MacOS/Navicat\ Premium\ Essentials
Navicat for Oracle	env TNS_ADMIN=~/Library/Containers/com.prect.NavicatORA/Data/Library/Application\ Support/PremiumSoft\ CyberTech/ DYLD_LIBRARY_PATH=~/Library/Containers/com.prect.NavicatORA/Data/Library/Application\ Support/PremiumSoft\ CyberTech/OCI arch -i386 /Applications/Navicat\ for\ Oracle.app/Contents/MacOS/Navicat\ for\ Oracle
Navicat Essentials for Oracle	env TNS_ADMIN=~/Library/Containers/com.prect.NavicatEssentialsForOracle/Data/Library/Application\ Support/PremiumSoft\ CyberTech/ DYLD_LIBRARY_PATH=~/Library/Containers/com.prect.NavicatEssentialsForOracle/Data/Library/Application\ Support/PremiumSoft\ CyberTech/OCI arch -i386 /Applications/Navicat\ Essentials\ for\ Oracle.app/Contents/MacOS/Navicat\ Essentials\ for\ Oracle

Before running the command, you need to copy the tnsnames.ora file and instant client to:

Navicat Version	tnsnames.ora	Instant Client
Navicat Premium	~/Library/Containers/com.prect.NavicatPremium/Data/Library/Application Support/PremiumSoft CyberTech/	~/Library/Containers/com.prect.NavicatPremium/Data/Library/Application Support/PremiumSoft CyberTech/OCI
Navicat	~/Library/Containers/com.prect.NavicatPremiumEssen	~/Library/Containers/com.prect.NavicatPremiumEssentials/

Premium Essentials	tials/Data/Library/Application Support/PremiumSoft CyberTech/	Data/Library/Application Support/PremiumSoft CyberTech/OCI
Navicat for Oracle	~/Library/Containers/com.prect.NavicatORA/Data/Library/Application Support/PremiumSoft CyberTech/	~/Library/Containers/com.prect.NavicatORA/Data/Library/Application Support/PremiumSoft CyberTech/OCI
Navicat Essentials for Oracle	~/Library/Containers/com.prect.NavicatEssentialsForOracle/Data/Library/Application Support/PremiumSoft CyberTech/	~/Library/Containers/com.prect.NavicatEssentialsForOracle/Data/Library/Application Support/PremiumSoft CyberTech/OCI

Commands (Available only in Full Version)

Navicat Premium

Navicat Objects	Server Type	Command Lines
Backup	MySQL, PostgreSQL , SQLite and MariaDB	<prog_name> --backup <connection-type> <connection-name> [<database-name> [<schema-name> [<profile-name>]]]
Restore Backup	MySQL, PostgreSQL, SQLite and MariaDB	<prog_name> --restore-backup <connection-type> <connection-name> <database-name> <schema-name> [<backup-name>]
Import	All	<prog_name> --import <connection-type> <connection-name> <database-name> <schema-name> <profile-name>
Export Table	All	<prog_name> --export <connection-type> <connection-name> <database-name> <schema-name> <profile-name>
Export View Result	All	<prog_name> --export-view <connection-type> <connection-name> <database-name> <schema-name> <profile-name>
Export Materialized View Result	Oracle	<prog_name> --export-mview oracle <connection-name> <schema-name> <profile-name>
Export Query Result	All	<prog_name> --export-query <connection-type> <connection-name> <database-name> <schema-name> <profile-name>
Query Execution	All	<prog_name> --query-execution <connection-type> <connection-name> <database-name> <schema-name> <query-name>
Data Transfer	All	<prog_name> --data-transfer <profile-type> < profile-name>
Data Synchronization	All	<prog_name> --data-synchronization <connection-type> <profile-name>
Batch Jobs	All	<prog_name> --batch-jobs <batch-job-name>
List Schedule	All	<prog_name> --list-schedule

Note:

<prog_name> - should be the path to the executable file inside: such as ./"Navicat Premium.app"/Contents/MacOS/"Navicat Premium"

<connection-type> - type of the connection: mysql, oracle, pgsql, sqlite, mssql or mariadb

<profile-type> - type of the data transfer profile: mysql, oracle, pgsql, sqlite, mssql, mariadb or premium

Navicat for MySQL/Navicat for MariaDB

Navicat Objects	Command Lines
Backup	<prog_name> --backup <connection-name> [<database-name> [<profile-name>]]
Restore Backup	<prog_name> --restore-backup <connection-name> <database-name> [<backup-name>]
Import	<prog_name> --import <connection-name> <database-name> <profile-name>
Export Table	<prog_name> --export <connection-name> <database-name> <profile-name>
Export View Result	<prog_name> --export-view <connection-name> <database-name> <profile-name>

Export Query Result	<prog_name> --export-query <connection-name> <database-name> <profile-name>
Query Execution	<prog_name> --query-execution <connection-name> <database-name> <query-name>
Data Transfer	<prog_name> --data-transfer <profile-name>
Data Synchronization	<prog_name> --data-synchronization <profile-name>
Batch Jobs	<prog_name> --batch-jobs <batch-job-name>
List Schedule	<prog_name> --list-schedule

Note:

<prog_name> - should be the path to the executable file inside: such as ./"Navicat for MySQL.app"/Contents/MacOS/"Navicat for MySQL"

Navicat for Oracle

Navicat Objects	Command Lines
Import	<prog_name> --import <connection-name> <schema-name> <profile-name>
Export Table	<prog_name> --export <connection-name> <schema-name> <profile-name>
Export View Result	<prog_name> --export-view <connection-name> <schema-name> <profile-name>
Export Materialized View Result	<prog_name> --export-mview <connection-name> <schema-name> <profile-name>
Export Query Result	<prog_name> --export-query <connection-name> <schema-name> <profile-name>
Query Execution	<prog_name> --query-execution <connection-name> <schema-name> <query-name>
Data Transfer	<prog_name> --data-transfer <profile-name>
Data Synchronization	<prog_name> --data-synchronization <profile-name>
Batch Jobs	<prog_name> --batch-jobs <batch-job-name>
List Schedule	<prog_name> --list-schedule

Note:

<prog_name> - should be the path to the executable file inside: such as ./"Navicat for Oracle.app"/Contents/MacOS/"Navicat for Oracle"

Navicat for PostgreSQL

Navicat Objects	Command Lines
Backup	<prog_name> --backup <connection-name> [<database-name> [<schema-name> [<profile-name>]]]
Restore Backup	<prog_name> --restore-backup <connection-name> <database-name> <schema-name> [<backup-name>]
Import	<prog_name> --import <connection-name> <database-name> <schema-name> <profile-name>
Export Table	<prog_name> --export <connection-name> <database-name> <schema-name> <profile-name>
Export View Result	<prog_name> --export-view <connection-name> <database-name> <schema-name>

	<profile-name>
Export Query Result	<prog_name> --export-query <connection-name> <database-name> <schema-name> <profile-name>
Query Execution	<prog_name> --query-execution <connection-name> <database-name> <schema-name> <query-name>
Data Transfer	<prog_name> --data-transfer <profile-name>
Data Synchronization	<prog_name> --data-synchronization <profile-name>
Batch Jobs	<prog_name> --batch-jobs <batch-job-name>
List Schedule	<prog_name> --list-schedule

Note:

<prog_name> - should be the path to the executable file inside: such as ./"Navicat for PostgreSQL.app"/Contents/MacOS/"Navicat for PostgreSQL"

Navicat for SQLite

Navicat Objects	Command Lines
Backup	<prog_name> --backup <connection-name> [<database-name> [<profile-name>]]
Restore Backup	<prog_name> --restore-backup <connection-name> <database-name> [<backup-name>]
Import	<prog_name> --import <connection-name> <database-name> <profile-name>
Export Table	<prog_name> --export <connection-name> <database-name> <profile-name>
Export View Result	<prog_name> --export-view <connection-name> <database-name> <profile-name>
Export Query Result	<prog_name> --export-query <connection-name> <database-name> <profile-name>
Query Execution	<prog_name> --query-execution <connection-name> <database-name> <query-name>
Data Transfer	<prog_name> --data-transfer <profile-name>
Data Synchronization	<prog_name> --data-synchronization <profile-name>
Batch Jobs	<prog_name> --batch-jobs <batch-job-name>
List Schedule	<prog_name> --list-schedule

Note:

<prog_name> - should be the path to the executable file inside: such as ./"Navicat for SQLite.app"/Contents/MacOS/"Navicat for SQLite"

Navicat for SQL Server

Navicat Objects	Command Lines
Import	<prog_name> --import <connection-name> <database-name> <schema-name> <profile-name>
Export Table	<prog_name> --export <connection-name> <database-name> <schema-name> <profile-name>
Export View Result	<prog_name> --export-view <connection-name> <database-name> <schema-name>

	<profile-name>
Export Query Result	<prog_name> --export-query <connection-name> <database-name> <schema-name> <profile-name>
Query Execution	<prog_name> --query-execution <connection-name> <database-name> <schema-name> <query-name>
Data Transfer	<prog_name> --data-transfer <profile-name>
Data Synchronization	<prog_name> --data-synchronization <profile-name>
Batch Jobs	<prog_name> --batch-jobs <batch-job-name>
List Schedule	<prog_name> --list-schedule

Note:

<prog_name> - should be the path to the executable file inside: such as ./"Navicat for SQL Server.app"/Contents/MacOS/"Navicat for SQL Server"

Hot Keys

Navicat Main Window

Keys	Action
OPTION-CMD-Z	Show Connection Tree
CMD-I	Object Information
CMD-1	Table
CMD-2	View
CMD-3	Function/SQLite Index
CMD-4	MySQL/MariaDB Event/SQLite Trigger
CMD-5	Query
CMD-6	Backup/Oracle Data Pump
CMD-7	Schedule
CMD-8	Model
CMD-L	Query Log
CTRL-SHIFT-# (# represents 0 to 9)	Open Object Window from Favorites List
OPTION-CMD-.	Close Database/Schema
CMD-.	Close Connection
CMD-Y	New Query
SHIFT-CMD-T	Data Transfer
SHIFT-CMD-C	Console
SHIFT-CMD-R	Restore Backup
CTRL-CMD-G	Setup Schedule
CTRL-CMD-O	Erase Schedule

Common

Keys	Action
CMD-N	New Object
CMD-D	Design Object
CTRL-SHIFT-RIGHT ARROW or CTRL-SHIFT-LEFT ARROW	Next Tab
CMD-0	Actual Size
CMD-+	Zoom In
CMD--	Zoom Out
CMD-F	Find Text
CMD-G	Find Next Text
SHIFT-CMD-G	Find Previous Text
OPTION-SHIFT-# (# represents 0 to 9)	Add to Favorites

SHIFT-CMD-D	Duplicate Object

Table Designer

Keys	Action
CMD-O	Open Table
CMD-DOWN ARROW	Add Insert
CMD-RIGHT ARROW	Insert Field
CMD-DEL	Delete Field
CMD-K	Set Field as Primary Key
SHIFT-CMD-N	Set Field to Allow NULL
CMD-F	Find Field
CMD-G	Find Next Field
SHIFT-CMD-G	Find Previous Field

Table Viewer/View Viewer

Keys	Action
CMD-UP ARROW	Import Wizard
CMD-DOWN ARROW	Export Wizard
SHIFT-CMD-F	Filter Wizard
OPTION-CMD-R	Apply Filter in Filter Wizard
SHIFT-CMD-V	Switch to Form View
CMD-+	Add Row
CMD--	Delete Row
SHIFT-CMD-N	Set NULL Value
CMD-B	Image
CMD-T	Text
CMD-J	Jump to Row
SHIFT-ARROW	Select Records

View/Query

Keys	Action
CMD-R	Run
SHIFT-CMD-R	Run Selected
SHIFT-CMD-E	Run a Statement from Here
CMD-O	Open View/Load Query
SHIFT-CMD-ENTER	Stop
OPTION-CMD-F	Find and Replace
CMD-E	Use Selection for Find

CMD-J	Jump to Selection
CMD-DOWN ARROW	Go to the End of Page

Oracle Debugger

Keys	Action
OPTION-CMD-ENTER	Run
SHIFT-CMD-O	Step Over
SHIFT-CMD-I	Step In
SHIFT-CMD-E	Step End
SHIFT-CMD-ENTER	Stop

Server Monitor

Keys	Action
CMD-E	End Process
SHIFT-CMD-R	Set Auto Refresh

Model

Keys	Action
CMD-D	Add Diagram in Model
SHIFT-CMD-P	Page Setup
CMD-P	Print
ESC	Select
H	Move Diagram
T	New Table
L	New Layer
N	New Note
I	New Image
R	New Relation
CMD-B	Bold Selected Table or Relation
CMD-0	Actual Size
CMD-+ or CMD-Mousewheel Up	Zoom In
CMD-- or CMD-Mousewheel Down	Zoom Out

Log Files

Navicat provides log files, namely **QueryExec.log** and **CmdLineJob.log**, to keep track on the actions that have been performed in Navicat and they are located in the default folder, e.g. ~/Library/Application Support/PremiumSoft CyberTech/Navicat Premium. You are allowed to change the log files location under Preferences.

QueryExec.log

Store all SQL statements of all the operations executed over databases and database objects in Navicat. Select **View** -> **Query Log** from the main menu or press CMD-L to open the QueryExec.log file.

CmdLineJob.log

Store information for Navicat command line process and all operations while running schedule.